**LAB REPORT**

***Submitted by***

**HARISH G [RA2111026010284]**

**NILAY KALE [RA2111026010287]**

**REETHIKA K [RA2111026010298]**

***Under the Guidance of***

**Dr. SASI REKHA SANKAR**

**Assistant Professor, CINTEL**

***In partial satisfaction of the requirements for the degree of***

## **BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**

**with specialization in Artificial Intelligence and Machine Learning**

## Logo, company name Description automatically generated

**SCHOOL OF COMPUTING**

# **COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**MAY 2023**

|  |  |
| --- | --- |
| Logo, company name  Description automatically generated | COLLEGE OF ENGINEERING & TECHNOLOGY  SRM INSTITUTE OF SCIENCE & TECHNOLOGY  S.R.M. NAGAR, KATTANKULATHUE – 603 203  Chengalpattu District |

**BONAFIDE CERTIFICATE**

Register No. RA2111026010284, RA2111026010288, RA2111026010298 Certified to be the bonafide work done by HARISH G, NILAY KALE, REETHIKA K of II Year/IV Sem B.Tech Degree Course in the **Practical Software Software Engineering and Project Management 18CSC206J** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Kattankulathur during the academic year 2022 – 2023.

**LAB INCHARGE Head of the Department**

**Ms. Sasi Rekha Shankar**

**Assistant Professor**

**CINTEL**

**SRMIST – KTR.**

**Date :**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | ABSTRACT |  |
|  | LIST OF FIGURES |  |
|  | LIST OF ABBREVIATIONS |  |
| 1 | PROBLEM STATEMENT |  |
| 2 | STAKEHOLDERS & PROCESS MODELS |  |
| 3 | IDENTIFYING REQUIREMENTS |  |
| 4 | PROJECT PLAN & EFFORT |  |
| 5 | WORK BREAKDOWN STRUCTURE & RISK ANALYSIS |  |
| 6 | SYSTEM ARCHITECTURE, USE CASE & CLASS DIAGRAM |  |
| 7 | ENTITY RELATIONSHIP DIAGRAM |  |
| 8 | DATA FLOW DIAGRAM |  |
| 9 | SEQUENCE & COLLABORATION DIAGRAM |  |
| 10 | DEVELOPMENT OF TESTING/USER INTERFACE |  |
| 11 | TEST CASES |  |
| 12 | ARCHITECTURE/DESIGN/IMPLEMENTATION |  |
|  | CONCLUSION |  |
|  | REFERENCE |  |
|  | APPENDIX(CODE) |  |

**ABSTRACT**

The Vehicle Maintenance System App is a mobile application designed to streamline the maintenance of vehicles for fleet managers and individual vehicle owners. The app allows users to schedule regular maintenance tasks, receive notifications for upcoming maintenance, and keep track of vehicle repair history. The app is user-friendly, easy to navigate, and can be customized to meet individual user needs. This project report provides a detailed overview of the development process, including the project requirements, design, implementation, testing, and deployment phases. The report also discusses the challenges encountered during the development process and the solutions adopted to overcome them. The app is expected to significantly improve vehicle maintenance efficiency, reduce costs, and increase overall vehicle performance. Overall, the Vehicle Maintenance System App is an innovative solution to simplify vehicle maintenance tasks and improve the reliability of vehicles.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure number** | **Title** | **Page number** |
| 2.1 | RAD prototype model |  |
| 4.1 | Project management |  |
| 4.2 | Roles and Responsibilities |  |
| 4.3 | Steps of a software change request |  |
| 5.1 | Work Breakdown Structure |  |
| 5.2 | Gantt Chart |  |
| 5.3 | SWOT Analysis |  |
| 6.1 | System Architecture Diagram |  |
| 6.2 | Use Case Diagram |  |
| 6.3 | Class Diagram |  |
| 7.1 | Entity Relation Diagram |  |
| 8.1 | Data Flow Diagram |  |
| 9.1 | Sequence Diagram |  |
| 9.2 | Collaboration diagram |  |
| 10.1 | Interface 1 |  |
| 10.2 | Interface 2 |  |
| 10.3 | Interface 3 |  |
| 10.4 | Interface 4 |  |
| 10.5 | Interface 5 |  |
| 10.6 | Interface 6 |  |
| 12.1 | Functional test case testing |  |
| 12.2 | Security testing |  |
| 12.3 | Alert counts by risks and confidence |  |
| 12.4 | Alerts counts by site and risk |  |
| 12.5 | Alerts |  |
| 12.6 | Alerts 2 |  |
| 12.7 | Alerts 3 |  |
| 12.8 | Home page |  |
| 12.9 | Add vehicle |  |
| 12.10 | Delete vehicle |  |
| 12.11 | Payment |  |
| 12.12 | Payment 2 |  |
| 12.13 | Schedule |  |
| 12.14 | Schedule 2 |  |
| 12.15 | Schedule 3 |  |
| 12.16 | History |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table number** | **Title** | **Page number** |
| 2.1 | Stakeholders |  |
| 10.1 | Methodologies |  |
| 11.1 | Functional test cases |  |
| 11.2 | Non-functional test cases |  |
| 12.1 | Manual testing report |  |

**PROBLEM STATEMENT**

**AIM:** To frame a project team, and analyse and identify a software project. To create a business case and arrive at a problem statement for the vehicle maintenance system.

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Register Number** | **Name** | **Role** |
| **1** | **RA2111026010287** | **Nilay Kale** | **Leader/Rep** |
| **2** | **RA2111026010284** | **Harish G** | **Member** |
| **3** | **RA2111026010298** | **Reethika K** | **Member** |

**Project Title: Vehicle Maintenance System**

**Project Description:**

The problem that the Vehicle Maintenance System App aims to address is the lack of a centralized, easy-to-use platform for vehicle owners and fleet managers to manage their vehicle maintenance tasks. Traditional methods of vehicle maintenance rely on manual record-keeping, which can be time-consuming, error-prone, and difficult to track. Moreover, vehicle owners often struggle to keep track of regular maintenance tasks, leading to increased repair costs and decreased vehicle performance. Additionally, fleet managers have the added challenge of managing multiple vehicles, making it even more challenging to keep track of maintenance tasks and repair history. The Vehicle Maintenance System App seeks to address these challenges by providing a user-friendly, centralized platform that simplifies vehicle maintenance tasks, reduces costs, and improves overall vehicle performance.

In this era of smart gadgets, smartphones are a crucial part of everyone’s daily life. Thus, by making an application that can be accessed anywhere using your mobile and Bluetooth, it becomes very convenient to maintain the condition of your vehicle during intensive daily use and be up to date with this modern smart software.

**Business Case**

**The Project:**

1. To make an application that can easily be accessed using your phone and Bluetooth.

2. To provide continuous updates on your vehicle’s condition.

3. Suggest repairs and allow you to make appointments regarding the same.

4. Keep track of the past history of your vehicle’s health.

5. Communicate with mechanics on things such as brake fluid, fuel efficiency, brake pads, tires’ condition, etc

**The History:**

To begin the competitive analysis, we selected the top 5 free applications in the App Store that offer some kind of car maintenance aid and interacted with them to test features such as the onboarding process, content strategy, customizability, data sync across multiple platforms, etc. We put myCARFAX, AUTOsist, Simply Auto, CarAssistant, and Drivvo to the test. We wanted to find out the scope and value of the aid they offered to users when it comes to taking care of routine car maintenance

**The Key takeaways from the analysis were:**

Confusing layouts, content strategy, and information architecture were common in 4 applications out of the 5 tested.

There were 3 gaps in terms of what these applications offered as a benefit for the user:

* The tested platforms did not allow mobile payments for services or scheduling appointments with various car service providers.
* Communication between the service providers and customers was not enabled or enhanced in any way.

Easy access to vehicles’ service histories was only allowed in 1 out of 5 applications.

**Limitations:**

* Security may become an issue for the success of the project, since the software contains details of the user profile, their vehicle details etc.
* People may not be able to use this system without having proper knowledge on using the software and internet facility
* The employees who work with the vehicle may not be experienced or may be in lack of special training.
* Reliability, there should be trustworthiness between the user and the system.

These limitations should be taken into consideration while creating software for the vehicle maintenance system.

**Approach:**

* Data Collection: Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.
* Security for mobile payments: Encryption is a security feature that uses the secret key to ensure private information is only accessible to the sending and receiving parties.
* Project requirements based on surveys: Using Questionnaires for Requirement Gathering. Requirements gathering is the process of collecting required project information from the client or customer before beginning the execution

**Benefits:**

The most prominent benefits of a vehicle management system include, but are not limited to the following:

**1. Vehicle status, readily available:**

To a driver, being notified about any issue, any indication of malfunction, or any sign that a driver forgot to check oil and fuel levels, lights and indicators, and tire pressure before leaving.

**2. Improved fuel efficiency:**

A VMS provides trip information such as fast acceleration, hard turning, harsh braking, speeding, and idling. Therefore, this type of information helps identify what needs to be addressed and how, on two different levels; driving behaviour and vehicle maintenance.

**3. Improved lifespan for vehicles and equipment:**

Mileage tracking in a trip log, fuel consumption tracking, diagnostic trouble code (DTC) alerts, and notifications about bad driving behaviour are bits of functionality that can help identify what needs to be dealt with and opt for repairs that are, in fact, required.

**4. Reduced labor costs:**

Automated, personalized reports of all sorts, driver scorecards, DTC alerts, and other notifications and mileage tracking with scheduled maintenance to go with it, can help drastically reduce labor costs.

**5. Improved capacity, more growth:**

Optimizing cost centres, and reducing waste and maintenance or labor costs can keep a fleet in order, but doubling its capacity invites new business and keeps customer retention to a maximum.

**Result:** Thus, the project team was formed, the project was described, the business case was prepared, and the problem statement was presented.

**PROCESS MODELS**

**AIM:** To identify the appropriate Process Model for the project and prepare Stakeholder and user Descriptions.

**Project Title: VehiCare**

**Selection of Methodology**

The Rapid Application Development (RAD) model is a process model that emphasizes rapid prototyping and iterative development. It is a suitable model for developing software systems with short development timelines and frequent changes in requirements. RAD model is particularly suitable for the development of the Vehicle Maintenance System (VMS) app due to the following reasons:

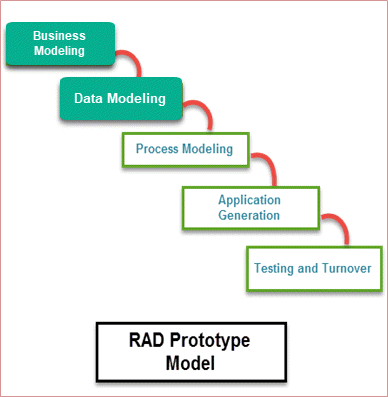


Fig 2.1 – RAD prototype model

**Rapid Prototyping:**

The RAD model allows for the development team to create quick prototypes of the VMS app. This is particularly useful for the VMS app because the app's interface needs to be intuitive and user-friendly. By using rapid prototyping techniques, the development team can quickly create and test different versions of the app's interface until it meets the end-users' expectations.

**Iterative Development:**

The RAD model's iterative development approach allows for frequent testing and feedback, which is essential for the VMS app. Since the VMS app is a critical system, it needs to be thoroughly tested to ensure that it is reliable, secure, and performs well. The RAD model's iterative development approach allows for frequent testing, which helps to identify and fix any issues in the app early in the development process.

**Collaborative Approach:**

The RAD model encourages collaboration between the development team and stakeholders, which is crucial for the VMS app. The VMS app is a system that involves different stakeholders, including drivers, fleet managers, and maintenance personnel. The RAD model's collaborative approach ensures that the app meets the needs and expectations of all stakeholders.

**Time-to-Market:**

The RAD model's rapid development approach allows for the VMS app to be developed and deployed quickly. This is particularly useful for the VMS app because it needs to be deployed as soon as possible to help manage and maintain the vehicle fleet. The RAD model's iterative development approach also allows for frequent releases of the app, which means that any new features or updates can be quickly rolled out to end-users.

In conclusion, the RAD model is a suitable process model for developing the Vehicle Maintenance System app due to its rapid prototyping, iterative development, collaborative approach, and quick time-to-market.

**STAKEHOLDERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stakeholder Name** | **Activity/ Area /Phase** | **Interest** | **Influence** | **Priority (High/ Medium/ Low)** |
| Govt. Agencies | Aiding in departments such as pollution control and road safety. | Low | High | 4 |
| Creditors | Extend money/resources for software development and testing. | High | High | 2 |
| Owners/ Car Entrepreneurs | Supply capital and have a say in how things are supposed to proceed. | High | High | 1 |
| Service Providers | Providing services like transport, store, prepare, inspect and repair vehicles. | High | High | 4 |
| Data Collectors | Data Collector responsibilities include downloading data, understanding survey objectives, and data analysis. | High | High | 3 |
| End Users | The end-user stakeholders are the people that will ultimately use your product, your services, or your solution and give feedback. | Low | High | 5 |
| Project Managers | Project manager is responsible for the planning, procurement, execution and completion of a project. He is accountable for team, failure & success of project | High | High | 2 |
| IT Department | The IT department helps lay down the ground rules for how people can use a company’s technology. It also involves creating, storing data, and assisting in the use of software throughout an organization. | High | High | 3 |
| Customers | Customers are people who pay money for a product or a service and may or may not end up using it. | High | Low | 6 |

Table 2.1 - Stakeholders

**Result:** Thus, the Project Methodology was identified and the stakeholders were described.

**REQUIREMENTS ENGINEERING:**

**AIM:** To identify the system, functional and non-functional requirements of the project.

**Functional Requirements:**

**Vehicle Management:** The system should allow fleet managers to manage the vehicles in their fleet, including adding new vehicles, updating vehicle information, and removing vehicles from the system.

**Maintenance Scheduling:** The system should allow fleet managers to schedule maintenance and repair work for their vehicles, including setting up alerts for routine maintenance and scheduling one-time repairs.

**Work Order Management:** The system should allow maintenance personnel to receive work orders and manage them, including updating the work order status, adding notes, and tracking progress.

**Vehicle Inspection Management:** The system should allow drivers to perform routine inspections of their vehicles, including reporting any issues or problems.

**Maintenance Reporting:** The system should allow fleet managers to generate reports on maintenance and repair work, including costs, time spent on repairs, and parts used.

**Integration with External Systems:** The system should allow integration with external systems, such as GPS tracking and fuel management systems, to provide real-time data on vehicle usage and fuel consumption.

**User Management:** The system should allow fleet managers to manage user accounts and access levels, including granting or revoking access to the system.

**Billing and Invoicing:** The system should allow the finance department to manage billing and invoicing, including generating invoices based on maintenance and repair work performed.

**Mobile Support:** The system should support mobile devices, allowing drivers and maintenance personnel to access the system from anywhere, anytime.

**Non-Functional Requirements:**

**Usability:** The system should have an intuitive and user-friendly interface that is easy to learn and use by drivers, maintenance personnel, and fleet managers.

**Reliability:** The system should be highly reliable and available at all times, with minimal downtime and disruption to the organization's operations.

**Performance:** The system should perform efficiently and quickly, with fast response times for all user interactions.

**Scalability:** The system should be scalable to accommodate future growth and expansion of the organization's fleet.

**Security:** The system should be highly secure, with strong authentication and authorization mechanisms, and encryption of sensitive data.

**Compatibility:** The system should be compatible with a wide range of hardware and software platforms, including mobile devices, GPS tracking systems, and fuel management systems.

**Maintainability:** The system should be easy to maintain and update, with a modular and flexible architecture that allows for easy changes and modifications.

**Interoperability:** The system should be able to integrate with other systems and software, including third-party applications and APIs.

**Performance under Load:** The system should be able to handle heavy user loads, especially during peak usage periods, without degradation in performance.

**Compliance:** The system should comply with relevant regulations and standards, such as those related to data privacy and security.

**Elicitation Techniques:**

**Survey:**

We conducted online surveys among various personal Drivers to analyse their common problems in maintaining their personal vehicles and their thoughts and ideas about it. We physically interacted with the common public about the issues they are facing, especially how they encounter their vehicle service timeline phases.

**Focus Group:**

In order to elicit more elaborate discussions on the topic of personal vehicle maintenance, a focus group with participants of the survey was conducted. A group of 5-6 personal drivers were selected for this focus group.

**FOCUS GROUP REPORT**

**COMPANY NAME:** VehiCare

**Date:** 07-02-2023

**INTRODUCTION**

In order to elicit more elaborate discussions on the topic of personal vehicle maintenance, a focus group with participants of the survey were conducted. A group of 5-6 personal drivers were selected for this focus group. The discussion was designed to gather information from the personal drivers.

**EXECUTIVE SUMMARY**

**1. Background:**

This report summarizes focus group findings conducted as part of a vehicle maintenance system. Findings are based on a focus group that was conducted. Focus groups explored people’s experiences with care and the ways in which their beliefs and values were, and were not, incorporated throughout the care process.

**2. Method:**

To begin the data analysis process, the survey was conducted prior to the focus group. This survey and focus group were conducted among the personal drivers and the collected data were used to analyse the publics’ problems and their perceptions about the idea of our project.

**3. Data analysis:**

The moderator reviewed the typed transcripts and written notes to ensure accuracy. The data were analyzed initially by looking for themes and variations in the comments from participants. Key Issues had been previously identified by the researchers. In addition, the participant’s comments were analyzed for their similarity or disparity with the comments of their employers. The research team members detected enough similarity of comments in the FGD to determine that saturation was achieved. Further analysis will be conducted with a programmed software package for qualitative data analysis.

**CHALLENGES**

The following challenges were identified by participants:

* It is difficult to maintain important information in books.
* More manual hours need to generate required reports.
* It is tedious to manage historical data which needs much space to keep all the previous years ledgers, books, etc.
* Daily sales and purchase details must be entered into books and are very difficult to maintain.

**OVERVIEW OF RESULT**

* Majority of participants accepted that this system will reduce their manual work and records.
* Most of the participants reported that they feel comfortable with the features proposed by us.
* Some of the participants suggested that they need to know the locations of service providers in unfamiliar areas too.
* All participants requested adding reminders for the service which is not present in any existing system.

**Result:** Thus, the requirements were identified and accordingly described.

**PROJECT PLAN AND EFFORT**

**AIM:** To prepare a project Plan on scope, Calculate Project effort based on resources, and Find Job roles and responsibilities.

**PROJECT PLAN**

**Integration Management**

**Governance Framework:**

The governance framework acts as an essential supporting structure, a framework of rules and practices by which the board ensures accountability, fairness, and transparency in how the company runs and communicates with its stakeholders.

The governance framework in project management typically includes the following components:

A picture containing timeline

Description automatically generated

Fig 4.1 – project management

**Project governance structure**: This includes the roles and responsibilities of project stakeholders, such as project sponsors, steering committees, project managers, and project teams. It outlines how decisions are made, who has the authority to make them, and how they are communicated.

**Project management processes**: This includes the processes and procedures for planning, executing, monitoring, and controlling projects. It outlines how project objectives are defined, how resources are allocated, and how project progress is monitored and reported.

**Project performance metrics**: This includes the measures used to evaluate project performance, such as cost, schedule, quality, and risk. It outlines how project performance is tracked and how corrective actions are taken when needed.

**Project management tools and technology**: This includes the tools and techniques used to manage projects, such as project management software, communication tools, and collaboration platforms. It outlines how these tools are used to facilitate project management processes.

**Project management standards and best practices**: This includes the standards and best practices for project management, such as PMI's Project Management Body of Knowledge (PMBOK) and the PRINCE2 methodology. It outlines how these standards and best practices are applied to projects within the organization.

**Project Team Structure, their roles, and Responsibilities**

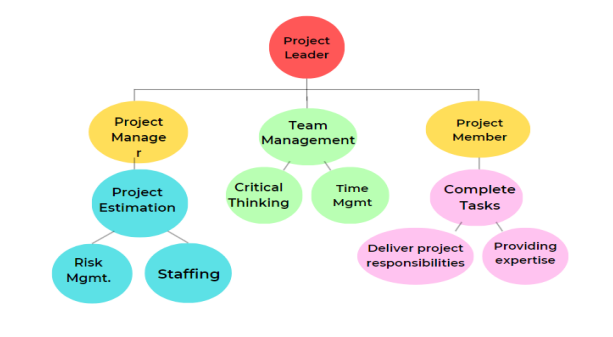
****

Fig 4.2 – roles and responsibilities

**Project Leader**

* Develop team schedules and assist in the successful onboarding and training of members
* Create and communicate a clear list of expectations and goals for team members to follow
* Offer emotional support to project team members and make people feel valued

**Project Manager**

* Planning everything from execution to delivery
* Directing the team to achieve a common goal
* Delegating work effectively
* Managing the resource time
* Managing documentation and reports

**Project Member**

* Complete the project objectives
* Members of a project team can have various roles.
* Delegating work effectively
* Work together to complete a project or goal

**COST MANAGEMENT**

**ESTIMATE EFFORT:**

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981. COCOMO is one of the most widely used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

**In COCOMO, projects are categorized into three types:**

1. Organic
2. Semi Detached
3. Embedded

**Organic:** A development project can be treated as organic if it deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.

Estimated Number of Lines of Code (SLOC): 2500 SLOC

**Effort = a(kLOC)b**

**Development Time = c(Effort)d**

Effort = 2.4(2.5)1.05

= 2.4(2.617)

= 6.281 persons per month

**Budget Control:**

Dev. Time = 2.5(6.28)0.38

= 2.5(2.01)

= 5.02 months

Controlling the budget in a vehicle maintenance system requires careful planning, monitoring, and management. Here are some key strategies to help you keep your project within budget:

Develop a detailed budget plan: Before the project begins, develop a detailed budget plan that outlines all the costs associated with the project. This plan should include all project costs, such as labor, hardware, software, training, travel, and any other expenses. The plan should be based on realistic estimates of the costs of the project and should be periodically reviewed and updated throughout the project.

**Monitor spending**: Monitor spending closely throughout the project to ensure that costs stay within budget. Track spending on a regular basis and compare actual costs to the budgeted costs to identify any areas where spending is exceeding the budget. This will enable the project manager to take corrective action to bring spending back within budget.

**Use cost-saving measures**: Implement cost-saving measures where possible to reduce project costs. This may involve negotiating with vendors for better pricing, using open-source software instead of commercial software, using cloud-based resources instead of on-premises resources, and minimizing travel and other expenses.

**Manage project scope**: Ensure that the project stays within scope and does not expand beyond the original requirements. Scope creep can cause costs to increase, so it is important to manage the scope carefully and to avoid adding unnecessary features or requirements to the project.

**Optimize resource utilization**: Optimize the utilization of resources, including labor, equipment, and materials, to ensure that they are being used efficiently and effectively. This may involve adjusting schedules or shifting resources to different areas of the project to avoid bottlenecks and delays.

**Use project management software**: Use project management software to track spending, manage resources, and monitor progress. This will enable the project manager to identify potential budget issues before they become significant problems and to take corrective action to keep costs under control.

**Regularly review and adjust the budget**: Regularly review and adjust the budget throughout the project to ensure that it remains accurate and realistic. This will enable the project manager to make adjustments to the budget as needed to accommodate changes in project scope, resource requirements, or other factors.

In summary, controlling the budget in a vehicle maintenance system requires a proactive and vigilant approach to planning, monitoring, and management. By developing a detailed budget plan, monitoring spending, using cost-saving measures, managing project scope, optimizing resource utilization, using project management software, and regularly reviewing and adjusting the budget, it is possible to keep costs under control and ensure the success of the project.

**CHANGE MANAGEMENT**

Change management in software development projects is the process of identifying, preparing, and supporting the implementation of software changes.

It is applied throughout the whole software development process. New requirements and the need for changes may appear suddenly and can shift several times. If you don’t manage them well, your project may be at risk of failing. Each software development project is defined by three dimensions: scope, time and budget. Most project changes affect some or all of these values. Also, every change poses a certain risk to the implementation of the software project you are developing. You can avoid or minimize risk but sometimes you just have to accept it. The key is to be aware of the full implications of each change for the software project you are working on. To deal with change management in the best possible way, stick to a consistent change management plan. Some modules/ prerequisites cannot be changed in our software are:

* User Login
* Booking Appointment
* Payment Gateway
* Car Details



Fig 4.3 – steps of a software change request

**PROJECT CLOSURE:**

Project closure is the critical last phase in the project management lifecycle.

During project closure, the team reviews the deliverables, then compares and tests its quality to the intended project outcome. Then they share the deliverables with the project’s client. It includes:

* Software Code
* Demo S/W
* Executable File
* Customer Report
* Sample Dataset
* Instructions
* Documentation

Effective project closure helps define a team’s and an organization’s culture. Approaching project closure as fundamental to building success leads to better-quality work and makes clients happy. Learning is a critical outcome of project closure. The process sets the stage for more successful and satisfying projects in the future.

**SCHEDULE MANAGEMENT:**

**1.Defining Milestones:**

A milestone is a specific point within a project’s life cycle used to measure the progress toward the ultimate goal. Milestones in project management are used as signal posts for a project's start or end date, external reviews or input, budget checks, submission of a major deliverable, etc. A milestone is a reference point that marks a significant event or a branching decision point within a project. For our projects we have defined various tasks/ modules and milestones to be achieved in between. Some of the tasks have already been completed while some are ongoing.

**2. Schedule Control:**

Schedule control is used in project management to monitor the activities and tasks to ensure you’re proceeding as planned. Of course, it’s more than just monitoring status. Project schedule control also means updating your project processes and managing change. The most common way to do this is by creating a baseline. A baseline represents how you expect the project to proceed. When you have a schedule baseline, you can look at your actual progress and compare it to the planned progress, which lets you know if you’re ahead or behind schedule. This whole process is simplified with the help of project scheduling software.

Project schedule control is essential because it is used to monitor and control the progress of the project. A project that misses its deadline is not a success. While there are many metrics to measure the success of a project, keeping to the schedule is among the most fundamental. Another important aspect of schedule control is that it manages the expectations of your project stakeholders. It keeps them informed of changes and how they will impact the project.

Furthermore, it tells the stakeholders how you’re going to respond to issues to get the project back on track. This process is key to keeping stakeholders in the loop and should be done throughout the life cycle of the project. Project planning, scheduling and control work hand-in-glove to reduce the impact on your budget. The benefit is financial stability, which projects need to deliver at a cost that has been approved by stakeholders. It also creates documentation that can be helpful when researching historical data for future projects similar to the one you’re managing currently.

**RESOURCE MANAGEMENT**

**People and skills:**

Developing vehicle maintenance software can be a complex process that requires a range of skills and expertise. Here are some of the people and skills that may be required to create vehicle maintenance software:

**1. Knowledge of vehicle maintenance processes:** In order to create effective vehicle maintenance software, you will need to have a good understanding of the maintenance processes used in the automotive industry. This includes knowledge of common maintenance tasks, recommended maintenance schedules, and best practices for vehicle care.

**2. Programming skills:** You will need strong programming skills in order to create software that is functional, user-friendly, and reliable. The programming languages commonly used in software development include Java, Python, C++, and C#.

**3. Knowledge of database design:** Vehicle maintenance software will need to store data such as vehicle information, service history, and maintenance schedules. You will need to be familiar with database design principles and tools such as SQL.

**4. Collaboration and communication:** You will likely be working as part of a team, so good collaboration and communication skills are essential. This includes the ability to work well with others, communicate effectively, and be open to feedback and suggestions.

**5. Project Managers:** They oversee the development of the software, ensuring that it is delivered on time and within budget.

**6. Technical Writers:** They create documentation that explains how to use the software and provide users with a guide to troubleshooting common issues.

**7. Quality Assurance (QA) Testers:** They test the software to identify and resolve any bugs, issues, or errors.

**Budget Required:**

The cost of developing vehicle maintenance software can vary depending on various factors such as the complexity of the software, features and functionalities required, development platform, and the hourly rate of the development team. It is difficult to provide a specific cost without more information about the project requirements.

However, to give you a general idea, the development of vehicle maintenance software in India may cost anywhere from INR 2,00,000 to INR 10,00,000 or more. This is just an estimate, and the actual cost may be higher or lower depending on the factors mentioned above. It's recommended to consult with a software development company to get a more accurate estimate based on your specific requirements.

**Facilities Required:**

Making vehicle maintenance software would require several facilities to ensure that the software functions as intended and provides a high-quality user experience. Here are some of the key facilities that may be required:

**1. Development Environment:** Developers will need access to a development environment that includes a code editor, debugging tools, and access to necessary libraries and APIs to develop the software.

**2. Database Management System:** A database management system is required to store data related to vehicle maintenance, such as maintenance schedules, vehicle details, and service records.

**3. Testing Environment:** A testing environment is required to ensure that the software functions correctly and is free of bugs. This may include tools for automated testing and manual testing.

**4. Security Measures:** Security measures should be implemented to protect the sensitive data stored in the software, such as user login information and vehicle maintenance records.

**5. Technical Support**: Technical support should be available to users to provide assistance with any issues they may encounter while using the software.

Overall, making vehicle maintenance software requires a combination of development, design, testing, deployment, and support facilities to ensure that the software is functional, user-friendly, and secure.

**Result:** Thus, the project plan was implemented successfully.

**WORK BREAKDOWN STRUCTURE AND RISK ANALYSIS**

**AIM:** To prepare Work breakdown structure, Timeline Chart and Risk identification table

**WORK BREAKDOWN STRUCTURE:**

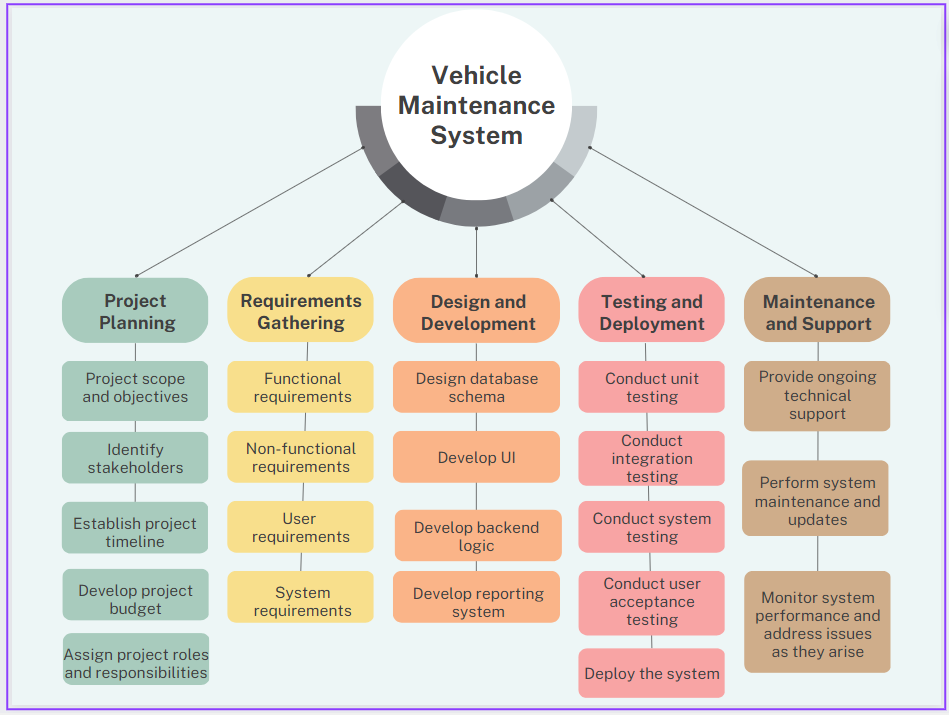


Fig 5.1 – work breakdown structure

**I. Project Planning**

A. Define project scope and objectives

B. Identify stakeholders

C. Establish project timeline

D. Develop project budget

E. Assign project roles and responsibilities

**II. Requirements Gathering**

A. Define functional requirements

B. Define non-functional requirements

C. Define user requirements

D. Define system requirements

**III. Design and Development**

A. Design database schema

B. Develop user interface

C. Develop backend logic

D. Develop reporting system

**IV. Testing and Deployment**

A. Conduct unit testing

B. Conduct integration testing

C. Conduct system testing

D. Conduct user acceptance testing

E. Deploy the system

**V. Maintenance and Support**

A. Provide ongoing technical support

B. Perform system maintenance and updates

C. Monitor system performance and address issues as they arise

**TIMELINE CHART**

Timeline charts are highly versatile visual charts that are used to illustrate a set of events chronologically. They're an excellent tool for conceptualizing event sequences or processes to gain insights into the nuances of a project. That could include summarizing historical events, or any other time frame where you need to measure minutes, hours, dates, or years. A timeline is a chart that depicts how a set of resources are used over time. If you're managing a software project and want to illustrate who is doing what and when, or if you're organizing a conference and need to schedule meeting rooms, a timeline is often a reasonable visualization choice. One popular type of timeline is the Gantt chart.

**Timeline

Description automatically generated with medium confidence**

Fig 5.2 – Gantt chart

**RISK MANAGEMENT**

**SWOT ANALYSIS:**

SWOT analysis is a useful tool for analyzing the strengths, weaknesses, opportunities, and threats of a product, service, or organization. In the case of vehicle maintenance software, a SWOT analysis might look like this:

**Strengths:**

**1.Automation:** The software can automate many tasks, such as scheduling maintenance and generating reports, which can save time and reduce errors.

**2.Customization:** The software can be customized to meet the specific needs of different users, such as fleet managers or individual car owners.

**3.Data management:** The software can store data on maintenance history, repair costs, and other information that can help users make informed decisions about vehicle maintenance.

**4.User-friendly interface:** The software can be designed with an intuitive interface that makes it easy to use, even for people with little technical knowledge.

**Weaknesses:**

**1.Cost:** The software can be expensive to purchase and maintain, which can be a barrier for small businesses or individual users.

**2.Technical issues:** The software may have technical glitches, bugs, or compatibility issues with other software, which can cause frustration and delays.

**3.Dependence on technology:** The software relies on technology, such as computers and internet connectivity, which can be vulnerable to outages or cybersecurity threats.

**Opportunities:**

**1.Growing market:** As more and more people rely on vehicles for transportation, the demand for vehicle maintenance software is likely to increase.

**2.Expansion into related markets:** The software could be adapted to other related markets, such as trucking, aviation, or marine industries.

**3.Integration with other software:** The software could be integrated with other software, such as GPS tracking or fuel management software, to offer a more comprehensive solution.

**Threats:**

**1.Competition:** There are already many companies offering vehicle maintenance software, which means the market may be crowded and competitive.

**2.Rapidly changing technology:** The software may become outdated quickly as new technologies emerge, which could make it less valuable over time.

**3.Economic downturns:** A downturn in the economy could reduce demand for software as companies and individuals cut back on spending.

Chart, bubble chart

Description automatically generated

Fig 5.3 – SWOT Analysis

**Result:** Thus, work breakdown structure and risk analysis is successfully done.

**SYSTEM ARCHITECTURE, USE CASE AND CLASS DIAGRAM**

**AIM:** Design a system Architecture, Use Case and Data Flow Diagram.

**SYSTEM ARCHITECTURE:**

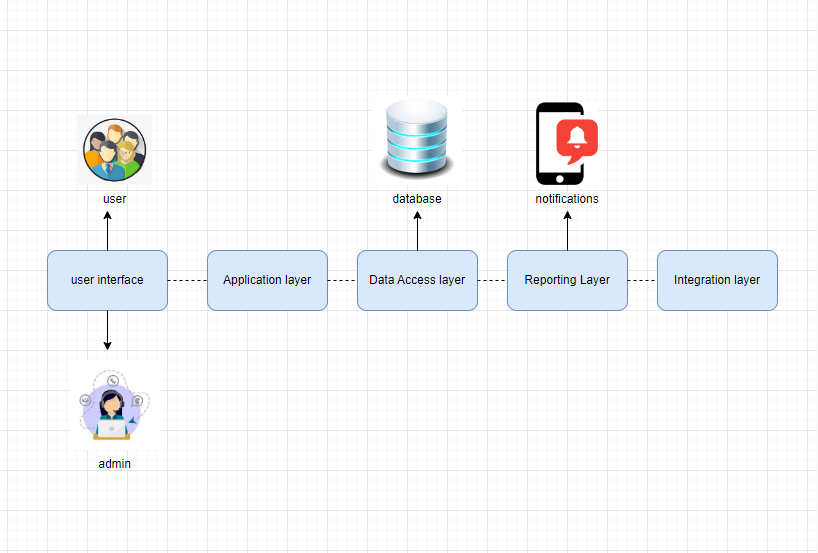
****

Fig 6.1 – system architecture diagram

The system architecture diagram is an abstract depiction of the system's component architecture. It provides a succinct description of the system's component architecture in order to assist in component-component relationships and system functioning.

It's a crucial tool that provides a comprehensive overview of the software system's physical deployment and development roadmap. An architectural diagram must perform a variety of tasks.

**1.User Interface:** This layer provides the graphical user interface(GUI) for the software. It allows users to interact with the system and input data.

**2.Application layer:** This layer contains business logic of the system. It processes user information and performs calculations and operations.

**3.Data Access layer:** This layer provides access to the database where all the relevant information about the vehicles and maintenance records is stored. It includes a mechanism to add, delete, and update the records.

**4.Database:** This layer stores all the information about the vehicles and their maintenance records.

**5.Reporting layer:** This layer generates reports based on the data stored in the database. It provides insights and analysis of the maintenance records.

**6.Integration layer:** This layer connects the vehicle maintenance software to the other systems such as external databases of APIs to exchange data and functionality.

**USE CASE DIAGRAM:**

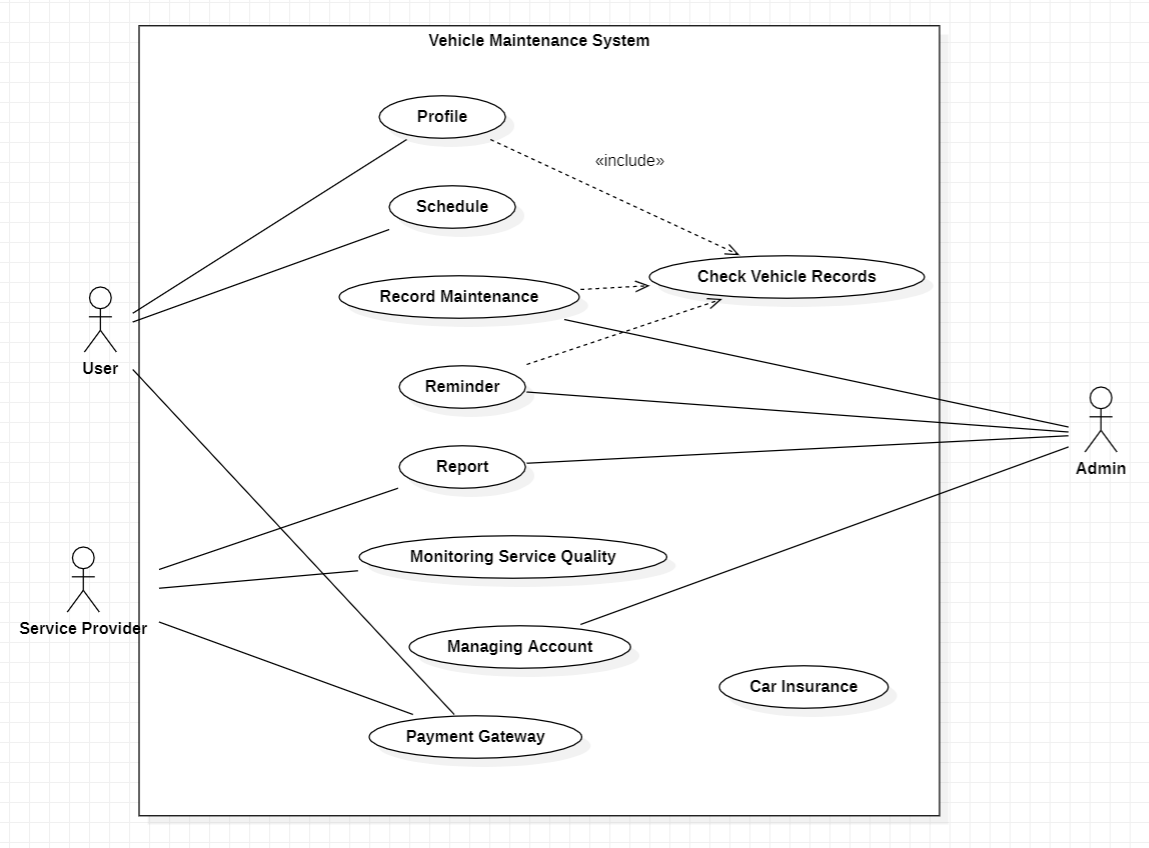
****

Fig 6.2 – Use Case diagram

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

**CLASS DIAGRAM**

**Diagram

Description automatically generated**

Fig 6.3 - class diagram

In this diagram, we have four classes: Vehicle, Service, Maintenance Record, and Maintenance Service. The **Vehicle** class has properties such as an ID, model, year, and mileage. The **Service** class represents a specific service performed on a vehicle and has properties such as an ID, date, the Vehicle object that the service was performed on, and the ServicePackage object that was used to perform the service.

The **MaintenanceRecord** class has properties such as an ID, date, type, and description. The **MaintenanceService** class represents the association between a Vehicle and a MaintenanceRecord and has properties such as an ID, vehicle ID, record ID and service date. The dependencies and generalization of classes are represented pictorially by connections between classes.

**Result:** Thus the system architecture, usecase and class diagram are successfully done.

**ENTITY REALTION DIAGRAM**

**AIM:** Design a Class, Entity Relation and Relational Diagram.

**ENTITY RELATION DIAGRAM**

**Diagram

Description automatically generated**

Fig 7.1 – entity relationship diagram

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.

Here there are 8 entities:  login, maintenance record, service record, user, customer, vehicle, vehicle maintenance, and schedule.

The **login** entity represents the login information such as username and password.

The **maintenance record** entity represents the record of the maintenance work done at present and has properties such as id, date, type, and description.

The **service record** entity represents the past service records of the vehicle and has properties such as id, date, service center, description,, and cost.

The **user** entity has the user information such as login id, name, mobile number, email and his login password.

The **customer** entity also represents the customer information and has properties like his id, password, name, mobile number and email address.

The **vehicle** entity represents the information of the vehicle and has properties like vehicle id, model, year of purchase, and mileage.

The **vehicle maintenance** entity holds the information of the current maintenance work and has properties such as vehicle id and record id.

The **schedule** entity represents the schedule of service appointments and has properties such as schedule id, dates and description.

**Result:** Thus entity relational diagram is successfully created.

**DATA FLOW DIAGRAM**

**AIM:** Design a state chart diagram

**DATA FLOW DIAGRAM:**

**Diagram

Description automatically generated**

Fig 8.1 – data flow diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences.

Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

**1. External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

**2. Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Generate Driver Report.”

**3. Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Driver Details.”

**4. Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Check Booking details.”

**Result:** Thus data flow diagram is successfully done.

**SEQUENCE AND COLLABORATION DIAGRAM**

**AIM:** Design a Sequence and Collaboration Diagram

**SEQUENCE DIAGRAM:**

A picture containing table

Description automatically generated

Fig 9.1 – sequence diagram

This is a Sequence Diagram for Vehicle Maintenance System, the sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. Its purpose is to model high-level interaction among active objects within a system. It models interaction among objects inside a collaboration utilizing a use case, it either models generic interactions or some certain instances of interaction.

The various Lifelines used here are:

* Logging in for the user
* Vehicle
* Scheduling Maintenance
* Checking Records

The various sub processes with their execution time have been depicted in the sequence diagram using lifelines and wait times.

**COLLABORATION DIAGRAM:**

**Diagram

Description automatically generated**

Fig 9.2 – collaboration diagram

The collaboration diagram comes under the UML representation which is used to visualize the organization of the objects and their interaction. Collaboration diagram is used to represent the structural organization of the system and the messages that are sent and received.It is better suited for depicting simpler interactions of smaller number of objects.

Here, the various objects used are:

* User
* Logging IN
* Vehicle Details
* Current Status
* Scheduling Maintenance
* Payment
* Checking Records

These objects interact with each other based on the requirements. The Messages sent and received are depicted using the arrows, between any two objects at a given point of time.

**Result:** Thus sequence and collaboration diagrams are successfully created.

**TESTING FRAMEWORK AND INTERFACE**

**AIM:** To develop a Testing Framework/User Interface

**Scope:**

The scope of a vehicle maintenance system project can vary depending on the specific requirements of the system. However, in general, the project scope would include the development of a system that can manage and track the maintenance activities of vehicles. This would involve creating a system that can store information about vehicles, such as their make, model, and year, as well as information about their maintenance history, such as when they were serviced and what repairs were made. The system would also need to include functionality for scheduling maintenance tasks, such as oil changes and tire rotations, and for tracking the progress of these tasks. In addition, the system would need to provide reports and analytics on maintenance activities, such as how often vehicles require maintenance and which types of maintenance are most common.

**Objective:**

The objective of a vehicle maintenance system project is to create a system that can effectively manage and track the maintenance activities of vehicles. The primary goal of the system is to help ensure that vehicles are kept in good working order and are safe to operate. This can help to minimize the risk of breakdowns and accidents, as well as reduce the costs associated with maintenance and repairs.The system should provide a centralized platform for managing all aspects of vehicle maintenance, from scheduling routine maintenance tasks to tracking the progress of repairs. By providing real-time visibility into maintenance activities, the system can help to ensure that maintenance tasks are completed on time and that any potential issues are identified and addressed promptly.

**Approach:**

* Begin with unit testing, which involves testing each component of the system individually to ensure that it is functioning correctly
* Use automated testing tools or manual testing methods for unit testing, depending on the complexity of the component and the resources available
* Subject the system to integration testing to test the interactions between different components of the system
* Identify any issues that may arise when different components are combined, such as conflicts between different data formats or compatibility issues with other systems
* Perform system testing to ensure that the system is functioning correctly as a whole and meets the requirements of the organization and its users
* Test the system's user interface, its functionality for managing maintenance tasks and schedules, and its reporting and analytics capabilities during system testing
* Subject the system to acceptance testing by testing it with real-world scenarios to ensure that it is meeting the needs of the organization and its users
* Test the system with a range of different vehicles and maintenance tasks during acceptance testing
* Document any issues that arise during testing and work with the development team to resolve them.

The testing phases for a vehicle maintenance system typically include the following:

**Unit Testing**: Unit testing is the first phase of testing, where individual components of the system are tested in isolation. This is done to ensure that each component is functioning as expected and to identify any defects early in the development cycle.

**Integration Testing**: In this phase, the individual components of the system are combined and tested together as a whole. This is done to ensure that the components are working together as expected and to identify any defects that may arise due to interactions between the components.

**System Testing**: System testing involves testing the entire system as a whole, including all the components and their interactions. This is done to ensure that the system meets the functional and non-functional requirements and to identify any defects that may have been missed in the previous testing phases.

**User Acceptance Testing**: User acceptance testing (UAT) is conducted to ensure that the system meets the user requirements and is easy to use. UAT is typically conducted by end-users or stakeholders who are familiar with the system requirements.

**Performance Testing**: Performance testing is done to ensure that the system meets the performance requirements, such as response time, throughput, and resource utilization. This is done to ensure that the system can handle the expected workload and can scale to meet future demand.

**Security Testing**: Security testing is done to ensure that the system is secure and protected from unauthorized access or attacks. This includes testing the system's access controls, encryption, and vulnerability assessments.

**Regression Testing**: Regression testing is conducted to ensure that the changes made to the system during the development cycle have not impacted the existing functionality of the system.

Overall, these testing phases are conducted in a systematic manner to ensure that the vehicle maintenance system is thoroughly tested and meets the requirements and expectations of the stakeholders.

**Testing Approach**

The testing approach for a vehicle maintenance system app would involve the following steps:

**Requirement Analysis**

The first step in testing the vehicle maintenance system app would be to analyze the requirements of the app to ensure that all the features and functionalities are identified and understood.

**Test Planning**

After analyzing the requirements, a test plan should be created that outlines the test objectives, scope, approach, and test schedule. The test plan should also identify the testing tools and techniques that will be used.

**Test Design**

In this phase, test cases are designed to validate the functionality and usability of the app. Test cases should be designed to test all the features of the app, including basic functionality, error handling, performance, and security.

**Test Execution**

Once the test cases have been designed, they should be executed to ensure that the app functions as expected. The test results should be documented, and defects should be logged in a defect tracking system.

**Test Reporting**

After executing the test cases, a test report should be created that summarizes the results of the testing. The report should include information on the test coverage, test results, and any defects that were found.

**Regression Testing**

Regression testing should be performed after any changes or updates are made to the app to ensure that existing functionalities are not impacted and that new functionalities are working as expected.

**User Acceptance Testing**

After the testing has been completed, the app should be tested by end-users to ensure that it meets their needs and is easy to use.

Overall, the testing approach for a vehicle maintenance system app should be comprehensive, covering all aspects of the app's functionality, performance, usability, security, and integration with other systems. The testing should be conducted in multiple phases and should involve the use of various testing tools and techniques to ensure that the app is thoroughly tested.

**Testing Environment:**

The testing environment for a vehicle maintenance system should be representative of the production environment in which the system will operate. This includes hardware, software, and network configurations that are similar to those in the production environment.

Here are some key components that should be included in the testing environment:

**Hardware**: The hardware should be similar to the production environment in terms of the type of computers, servers, and mobile devices that will be used by the end-users. This includes ensuring that the hardware has the required processing power, memory, and storage capacity to handle the application.

**Software**: The software used in the testing environment should be the same as that used in the production environment. This includes the operating system, database management system, web server, and any other software components used by the application.

**Network**: The network infrastructure used in the testing environment should be similar to the production environment in terms of the network topology, protocols, and bandwidth. This is important to ensure that the application performs well under realistic network conditions.

**Test Data**: The testing environment should have a representative sample of the data that will be used in the production environment. This includes customer data, transaction data, and any other data that is required to test the functionality of the application.

**Test Tools**: The testing environment should have all the required testing tools and software installed, including testing frameworks, test automation tools, and debugging tools.

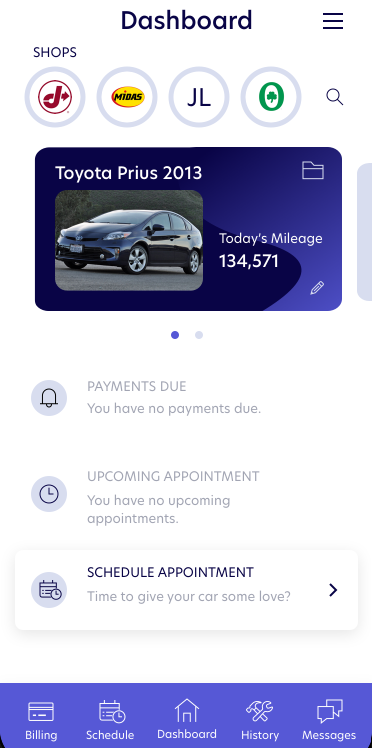
**Security**: The testing environment should be secured to ensure that the application is protected from unauthorized access or attacks. This includes securing the network, servers, and any other components that are part of the application.

Overall, the testing environment should be well-designed and maintained to ensure that the testing is effective and that the application is thoroughly tested before it is released into the production environment.

|  |  |  |
| --- | --- | --- |
| Types of Testing | Methodology | Tools Required |
| Functional Testing | Black Box Testing | Selenium, TestComplete, Appium |
| Performance Testing | Load Testing | JMeter, LoadRunner |
| Usability Testing | User-Centered Design | UsabilityHub, UserTesting.com |
| Security Testing | Penetration Testing | Burp Suite, OWASP ZAP |
| Integration Testing | Top-Down or Bottom Up | SOAPUI, Postman |
| Acceptance Testing | User Acceptance Testing | Manual Testing |
| Regression Testing | Automated Regression Testing | Selenium, TestComplete |

table 10.1 - methodologies

**Key Screen Interfaces**

**                Graphical user interface, application

Description automatically generated**

Fig 10.1 – interface 1 fig 10.2 – interface 2

**Graphical user interface, application

Description automatically generated        Graphical user interface, application

Description automatically generated**

Fig 10.3 – interface 3 fig 10.4 – interface 4

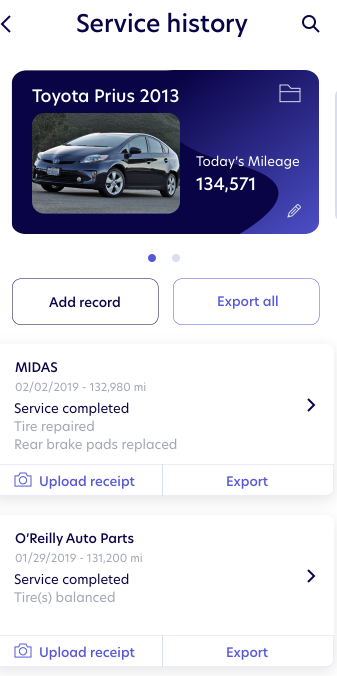
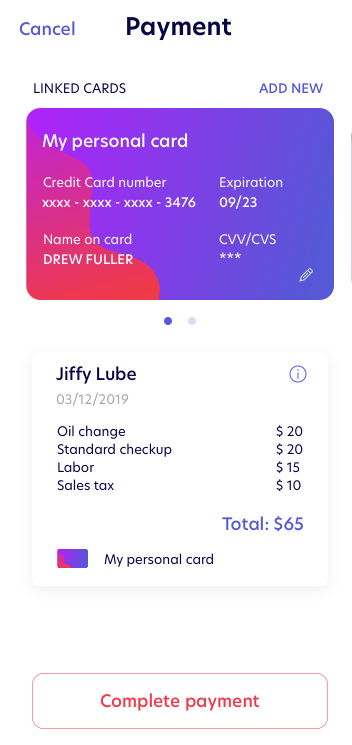
**             **

Fig 10.5 – interface 5 fig 10.6 – interface 6

**Result:** Thus the testing framework/user interface framework has been created for the vehicle maintenance system.

**TESTCASES**

**AIM:** To identify testcases

# **Functional Test Cases:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID (#)** | **Test Scenario** | **Test Case** | **Execution Steps** | **Expected Outcome** | **Actual Outcome** | **Status** |
| T01 | User Login/Sign up | Accept valid email id and password | 1. User clicks on User Registration link  2. Enter the email address  3. Enter the password | User should be taken to the next page for entering more user details | User taken to the next page for entering more details in case of sign in, and entered the home page in case of log in. | Passed |
| T02 | Adding/Editing  a new vehicle | Accept vehicle details | 1. User clicks on add vehicle Option.  2. User enters the details of the vehicle like image, model, mileage,etc. | The added vehicle should show in the home page.  If edited then the data should be updated. | The added vehicle is updated in the home page. The modified data also reflects in the home page. | Passed |
| T03 | Adding a Maintenance Task | Add the maintenance Task | 1.  1. User clicks on     Book service under a particular vehicle.  2.  2. The User chooses the service centre/shop.  3.  3. User will provide details for the service.  4.  4. User will choose the date and time of delivery of the vehicle. | The Booked status should be shown in appointments along with service details. | The booking page gets expected details from the user but the status showing part is still under development | Passed |
| T04 | Reminder | App should remind the user about upcoming services when it is due. | **–** | The due reminders should remind  in-app like in upcoming services | The reminders mention the due service in the Upcoming Service section. | Passed |
| T05 | View Service History | Verify that the user can view the maintenance history for each of their vehicles, including completed tasks. | The user clicks the option called History. | The history should show previous maintenance of the vehicle with vehicle details for each vehicle. | The past tasks of a service reflect in the history as a service is considered as completed. | Passed |
| T06 | Search Vehicles | Verify that users can search for a vehicle by make, model, or year | Users can search for a vehicle while entering new vehicle. | Users can search for a vehicle by entering the make, model, or year and the app displays all matching vehicles | Users can search for a vehicle by entering the make, model, or year and the app displays few matching vehicles | Partially Passed |

Table 11.1 - Functional testcases

# **Non-Functional Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID (#)** | **Test Scenario** | **Test Case** | **Expected Outcome** | **Actual Outcome** | **Status** |
| T07 | Security Testing | Verify that the app is secure and that user data is protected from unauthorized access or manipulation. | No security vulnerabilities found | 2 FireBase injection vulnerabilities found, both of which were fixed | Partially Passed |
| T08 | Performance Testing | Evaluate the app's performance under different conditions, such as high traffic or low network connectivity, to ensure that it is responsive and reliable. | App responds within 2 seconds under all conditions | App responds within 2 seconds under all conditions | Passed |
| T09 | Compatibility Testing | Test the app on different devices, operating systems to ensure that it works correctly and consistently across a range of platforms. | The app works consistently across all devices. | App doesn’t work on iOS devices. (Still under development) | Partially Passed |
| T10 | Usability Testing | Evaluate the app's usability and user experience, testing how easy it is for users to navigate the app and complete tasks. | Users can complete tasks easily and with minimal confusion | Users can complete tasks easily and with minimal confusion | Passed |
| T11 | Recovery Testing | Test the app's ability to recover from failures or crashes, such as by simulating a sudden loss of network connectivity. | App can recover from failures or crashes without data loss | App loses some data when the server crashes unexpectedly | Failed |
| T12 | User Acceptance Testing | This includes testing for factors such as user satisfaction, user engagement, and user feedback. | Users are satisfied with the app's functionality and usability | Users report dissatisfaction with the app's layout and some features | Partially Passed |

Table 11.2 – Non-functional testcases

**Result:** Thus, the test case manual has been created for the vehicle maintenance system.

**MANUAL TESTCASES TESTING REPORT**

**AIM:** Manual Test Case Reporting

Functional testing of a vehicle maintenance system includes testing the system's ability to perform specific functions such as creating and scheduling maintenance tasks, updating maintenance records, and generating reports. During functional testing, testers will perform a series of tests to verify that the system meets the functional requirements outlined in the software specifications. This may involve creating test cases, executing tests, and documenting any issues or bugs found during testing.

Non-functional testing of a vehicle maintenance system includes testing the system's performance, security, and usability. Performance testing may involve measuring the system's response time when executing tasks, testing its scalability and reliability under heavy loads, and verifying that it meets any performance requirements outlined in the software specifications. Security testing may involve verifying that the system's data is protected against unauthorized access and ensuring that the system adheres to any security regulations. Usability testing may involve testing the system's ease of use, accessibility, and user experience.

**OBSTACLES**

**Performance Testing:**

* **Limited Resources:** Conducting performance testing can require a significant amount of resources such as tools, test environments, and skilled testers. Limited resources can lead to incomplete or inaccurate testing.
* **Difficulty in Reproducing Issues:** Performance issues can be intermittent and challenging to reproduce. This can lead to missed defects and the inability to identify and fix issues.
* **Data Availability:** Performance testing requires a sufficient amount of data to test the system's performance accurately. However, it can be challenging to obtain and manage large amounts of data, leading to incomplete or inaccurate test results.

**Solution:**

* Allocate adequate resources such as skilled testers, suitable testing tools, and test environments.
* Conduct performance testing in a controlled environment that closely

matches the production environment.

* Use automated tools to generate load and monitor the system's response time, throughput, and resource utilization.

**User-Acceptance Testing:**

* **Limited User Participation:** User-acceptance testing requires user participation and feedback to be effective. However, it can be challenging to get sufficient user participation, leading to incomplete or inaccurate test results.
* **Limited Test Coverage:** User-acceptance testing may not cover all possible test cases, resulting in limited test coverage and the possibility of missing critical issues.
* **Human Error:** Manual user-acceptance testing involves human testers who are prone to making errors while executing test cases. This can result in false negatives or false positives and may lead to missed defects.

**Solution:**

* Involve a diverse set of users with varying levels of expertise to participate in testing.
* Use well-defined acceptance criteria and test scenarios to ensure comprehensive test coverage.
* Use automated tools to capture user feedback and track issues.

**Recovery Testing:**

* **Limited Test Coverage:** Recovery testing can be time-consuming and may not cover all possible test cases. This can result in limited test coverage and the possibility of missing critical issues.
* **Limited Resources:** Recovery testing requires specialized tools and resources to simulate failure scenarios and test the system's ability to recover from them. Limited resources can lead to incomplete or inaccurate testing.
* **Complexity:** Recovery testing can involve complex test scenarios that are challenging to create and execute. This can lead to inadequate test coverage, missed defects, and inaccurate test results.

**Solution:**

* Use automated tools to simulate failure scenarios and test the system's ability to recover from them.
* Test in a controlled environment that closely matches the production environment.
* Conduct a comprehensive risk analysis to identify potential failure scenarios and their impact.

**Security Testing:**

* **Complexity:** Security testing can involve complex test scenarios that are challenging to create and execute. This can lead to inadequate testncoverage, missed defects, and inaccurate test results.
* **Limited Test Coverage:** Security testing can be time-consuming, and it may not be possible to test all possible test cases. This can result in limited test coverage and the possibility of missing critical security issues.
* **Human Error:** Manual security testing involves human testers who are prone to making errors while executing test cases. This can result in false negatives or false positives and may lead to missed security defects.

**Solution:**

* Use industry-standard security testing frameworks and methodologies.
* Conduct a comprehensive risk analysis to identify potential security threats and vulnerabilities.
* Use automated tools to perform vulnerability scanning, penetration testing, and security assessments.

**TESTING PROCESS**

**Functional Test-Cases**

**Test Case Execution:** This step is to execute the designed functional test cases manually, step-by-step, to verify if the actual output of the application matched the expected output specified in the test case. During test case execution, the tester verifies all the functionalities of the software application, including the input validations, business logic, user interface, error messages, and other functional requirements.

**Defect Reporting:** During the test case execution, if the actual output does not match the expected output, the tester identifies the issue and reports it as a defect in a defect tracking tool. The defect report typically includes the steps to reproduce the issue, the expected behavior, the actual behavior, and any relevant screenshots or logs.

**Defect Verification**: After the defects are reported, the development team investigates the issue and fixes the defect. The tester then verifies the defect fix to ensure that the expected behavior is achieved.

**Non-Functional Test-Cases**

**Performance Testing:** In performance testing, test cases are designed to measure the speed, scalability, and stability of the software application under different load conditions. The test cases typically include scenarios such as measuring the response time of the application, testing the application under heavy load, and measuring the system resources used by the application under different load conditions.

**Compatibility Testing:** In compatibility testing, test cases are designed to ensure that the software application works correctly with different hardware, software, and operating system configurations. The test cases typically include scenarios such as testing the application on different browsers, testing the application on different devices, and testing the application on different operating systems.

**Integration Testing:** Integration testing is a type of testing that is performed to verify that different components of the software application work together correctly. In manual integration testing, the tester typically follows a test plan that outlines the steps needed to test the integration points between different components. The tester will perform the tests and record the results, looking for any issues or bugs that may arise when the components are integrated together.

**User Acceptance Testing (UAT):** User Acceptance Testing is a type of testing that is performed to ensure that the software application meets the user's requirements and expectations. In manual UAT testing, the tester typically follows a test plan that outlines the user scenarios to be tested. The tester will perform the tests and record the results, looking for any issues or bugs that may arise when the user scenarios are executed. The goal of UAT testing is to ensure that the software application is ready for release to the end-users.

**Recovery Testing:** Recovery Testing is a type of testing that is performed to verify that the software application can recover from various failures or crashes. In manual recovery testing, the tester typically follows a test plan that outlines the failure scenarios to be tested. The tester will perform the tests and record the results, looking for any issues or bugs that may arise when the application is recovering from a failure or crash. The goal of recovery testing is to ensure that the software application can recover gracefully and continue to function correctly after a failure or crash.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Progress Against Plan** | **Status** | **Software** |
| **Functional Testing** | **Green** | **Completed** |  |
| Log-in | **100%** | Completed | — |
| Add Vehicle | **100%** | Completed | — |
| Add Service | **100%** | Completed | — |
| View Service History | **100%** | Completed | — |
| Search Vehicles | **100%** | Completed | — |
| View Upcoming Services | **100%** | Completed | — |
| **Non-Functional Testing** | **Amber** | **In-Progress** |  |
| Performance Testing | **90%** | In-Progress | Selenium |
| Compatibility Testing | **100%** | Completed | Selenium |
| Integration Testing | **100%** | Completed | SoapUI |
| User- Acceptance Testing | **80%** | In-Progress | — |
| Security Testing | **75%** | In-Progress | OWASP ZAP |
| Recovery Testing | **75%** | In-Progress | OWASP ZAP |

Table 12.1 – Manual Testing Report

**Testing Software Report:-**

**Selenium: (Functional Test Case Testing)**

Graphical user interface, application

Description automatically generated

Fig 12.1 – functional test case testing

**OWASP ZAP: (Security Testing)**

Graphical user interface, text, application, email

Description automatically generated

Fig 12.2 – security testing

Table

Description automatically generated

Fig 12.3 – alert counts by risk and confidence

Timeline

Description automatically generated with medium confidence

Fig 12.4 – alert counts by site and risk

Graphical user interface, text, application, email

Description automatically generated

Fig 12.5 - alerts

Graphical user interface, text, application, email

Description automatically generated

Fig 12.6 – alerts 2

Table

Description automatically generated

Fig 12.7 – alerts 3

**Result:** Thus, the test case report has been created for the vehicle maintenance system.

**CODE WITH IMPLEMENTATION**

**AIM:** Code with implementation

**Main.dart**

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:vehicle\_maintenance\_app/checkscreen.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/ongenerateroute.dart';

import 'package:vehicle\_maintenance\_app/screens/addnewcar.dart';

import 'package:vehicle\_maintenance\_app/screens/loginpage.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/dashboard.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/homeparent.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/billingmain.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/paymentscreen.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleappointment.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleconfirmation.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/schedulereview.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleshop.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/schedulesuccess.dart';

import 'package:vehicle\_maintenance\_app/screens/signup.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/paymentsuccessfull.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:vehicle\_maintenance\_app/screens/upcoming\_appointments.dart';

import 'package:vehicle\_maintenance\_app/services/decision.dart';

String initialroute = '';

void main() async {

SystemChrome.setSystemUIOverlayStyle(SystemUiOverlayStyle(

statusBarColor: Colors.transparent,

));

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

initialroute = await logindecision();

runApp(MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

theme: ThemeData(

fontFamily: 'opensans',

useMaterial3: true,

scaffoldBackgroundColor: Colors.white,

colorSchemeSeed: maintheme,

elevatedButtonTheme: ElevatedButtonThemeData(

style: ElevatedButton.styleFrom(

surfaceTintColor: Colors.white,

foregroundColor: maintheme,

),

),

),

initialRoute: initialroute,

onGenerateRoute: RouteGenerator.generateRoute,

);

}

}

**addnewcar.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/data.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/loadingblock.dart';

class addNewCar extends StatefulWidget {

const addNewCar({Key? key}) : super(key: key);

@override

State<addNewCar> createState() => \_addNewCarState();

}

class \_addNewCarState extends State<addNewCar> {

String? carmaker;

String? carmodel;

UserServices userServices = UserServices();

@override

void initState() {

// TODO: implement initState

super.initState();

if (carmaker == null) {

carmaker = carMakers.keys.first;

carmodel = carMakers[carmaker][0];

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: IconButton(

onPressed: () {

Navigator.pop(context);

},

icon: Icon(

CupertinoIcons.left\_chevron,

color: darktext,

),

),

title: Text(

"Add Vehicle",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

actions: [

IconButton(

onPressed: () async {

loadingBlock(context: context);

await userServices.addnewcar(

carmaker: carmaker!,

carmodel: carmodel!,

);

Navigator.pop(context); // to pop dialog box

Navigator.pop(context); // to pop screen

print('Successfull');

},

icon: Icon(

Icons.check\_rounded,

color: darktext,

),

),

],

),

body: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 20,

),

Container(

height: 200,

width: MediaQuery.of(context).size.width,

margin: EdgeInsets.symmetric(vertical: 10),

child: ClipRRect(

borderRadius: BorderRadius.circular(16),

child: Image.asset(

'assets/images/vehicles/' + carPhotos[carmodel!].toString(),

fit: BoxFit.cover,

),

),

// child: Text(carmodel!),

decoration: BoxDecoration(

color: Colors.black.withAlpha(50),

borderRadius: BorderRadius.circular(16),

),

),

SizedBox(

height: 20,

),

Text(

'Vehicle Maker',

style: subtitle,

),

SizedBox(

height: 15,

),

Container(

padding: EdgeInsets.symmetric(horizontal: 15),

decoration: BoxDecoration(

borderRadius: BorderRadius.circular(12),

border: Border.all(

width: 1,

color: maintheme,

),

),

child: DropdownButton(

value: carmaker,

alignment: AlignmentDirectional.centerStart,

borderRadius: BorderRadius.circular(12),

isExpanded: true,

style: TextStyle(

fontSize: 16,

color: darktext,

),

underline: Container(),

icon: Icon(

CupertinoIcons.chevron\_down,

size: 20,

color: darktext,

),

items: [

for (String i in carMakers.keys)

DropdownMenuItem(

value: i,

child: Text(i),

),

],

onChanged: (a) {

setState(() {

carmaker = a;

carmodel = carMakers[carmaker][0];

});

},

),

),

SizedBox(

height: 20,

),

Text(

'Vehicle Model',

style: subtitle,

),

SizedBox(

height: 15,

),

Container(

padding: EdgeInsets.symmetric(horizontal: 15),

decoration: BoxDecoration(

borderRadius: BorderRadius.circular(12),

border: Border.all(

width: 1,

color: maintheme,

),

),

child: DropdownButton(

value: carmodel,

alignment: AlignmentDirectional.centerStart,

borderRadius: BorderRadius.circular(12),

isExpanded: true,

style: TextStyle(

fontSize: 16,

color: darktext,

),

underline: Container(),

icon: Icon(

CupertinoIcons.chevron\_down,

size: 20,

color: darktext,

),

items: [

for (String i in carMakers[carmaker])

DropdownMenuItem(

value: i,

child: Text(i),

),

],

onChanged: (a) {

setState(() {

carmodel = a;

});

},

),

),

SizedBox(

height: 30,

),

],

),

),

);

}

}

**billingmain.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/paymentwidgets/paymentduesection.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/paymentwidgets/paymenttiles.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/carcarousal.dart';

class billingMain extends StatefulWidget {

const billingMain({Key? key}) : super(key: key);

@override

State<billingMain> createState() => \_billingMainState();

}

class \_billingMainState extends State<billingMain> {

UserServices userServices = UserServices();

List<String> carkeys = [];

bool loaded = false;

int currentpage = 0;

void setcurrentpage(int page) {

print(page);

setState(() {

currentpage = page;

});

}

Future<QuerySnapshot> getcardata() async {

QuerySnapshot data = await userServices.getcars();

carkeys.clear();

for (DocumentSnapshot snapshot in data.docs) {

carkeys.add(snapshot.id.toString());

}

loaded = true;

return data;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

automaticallyImplyLeading: false,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

title: Text(

"Billing",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

child: FutureBuilder(

future: getcardata(),

builder: (context, snapshot) {

if (snapshot.hasData && snapshot.data!.size != 0) {

return Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 15,

),

Container(

height: 180,

child: carCarousal(

setCurrentpage: setcurrentpage,

items: [

for (DocumentSnapshot doc in snapshot.data!.docs)

buildVehiclecard(

carmaker: doc.get('carmaker'),

carmodel: doc.get('carmodel'))

],

),

),

SizedBox(

height: 15,

),

Expanded(

child: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: SingleChildScrollView(

physics: BouncingScrollPhysics(),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: [

paymentDueSection(

carkey: carkeys[currentpage],

),

SizedBox(

height: 15,

),

Text(

'RECENT TRANSACTIONS',

style: subtitle.copyWith(

color: darktext,

),

),

SizedBox(

height: 5,

),

recentTransactionsTile(),

],

),

),

),

),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

TextButton(

onPressed: () {},

child: Text('See all Transactions'),

),

],

),

],

);

} else if (snapshot.hasData && snapshot.data!.size == 0) {

return Center(

child: Text(

'Please add Vehicle in DashBoard',

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: darktext,

),

),

);

}

return Center(

child: CircularProgressIndicator(),

);

},

),

),

);

}}

**carcarousal.dart**

import 'dart:ui';

import 'package:carousel\_slider/carousel\_slider.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/data.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

class carCarousal extends StatefulWidget {

final Function? setCurrentpage;

final List items;

const carCarousal({Key? key, this.setCurrentpage, this.items = const []})

: super(key: key);

@override

State<carCarousal> createState() => \_carCarousalState();

}

class \_carCarousalState extends State<carCarousal> {

int currentpage = 0;

@override

Widget build(BuildContext context) {

return Column(

children: [

Expanded(

child: Container(

// height: 180,

width: MediaQuery.of(context).size.width,

child: CarouselSlider(

items: [

for (var i in widget.items) i,

],

options: CarouselOptions(

enableInfiniteScroll: false,

viewportFraction: 16 / 9,

enlargeFactor: 0.3,

enlargeCenterPage: true,

initialPage: 0,

scrollPhysics: BouncingScrollPhysics(),

onPageChanged: (int page, reason) {

if (widget.setCurrentpage != null) {

widget.setCurrentpage!(page);

}

setState(() {

currentpage = page;

});

}),

),

),

),

SizedBox(

height: 10,

),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

for (int i = 0; i < widget.items.length; i++)

Row(

children: [

AnimatedContainer(

duration: Duration(milliseconds: 600),

curve: Curves.fastLinearToSlowEaseIn,

height: 7,

width: (i == currentpage) ? (20) : (7),

decoration: BoxDecoration(

color: (i == currentpage)

? (maintheme)

: (Colors.grey.withAlpha(150)),

borderRadius: BorderRadius.circular(30),

),

),

SizedBox(

width: 5,

),

],

),

],

),

],

);

}

}

class buildVehiclecard extends StatelessWidget {

final String carmaker;

final String carmodel;

const buildVehiclecard(

{Key? key, required this.carmaker, required this.carmodel})

: super(key: key);

@override

Widget build(BuildContext context) {

print(carPhotos[carmodel].toString());

return ClipRRect(

borderRadius: BorderRadius.circular(16),

child: Container(

height: 180,

width: MediaQuery.of(context).size.width - 30,

decoration: BoxDecoration(

color: Colors.black,

),

child: Stack(

children: [

Container(

height: 180,

width: MediaQuery.of(context).size.width,

decoration: BoxDecoration(

image: DecorationImage(

fit: BoxFit.fill,

image: AssetImage(

'assets/images/vehiclebg.jpg',

),

),

),

child: BackdropFilter(

filter: ImageFilter.blur(

sigmaX: 2,

sigmaY: 2,

),

child: Container(

decoration: BoxDecoration(

color: Colors.black.withAlpha(25),

),

),

),

),

Positioned(

child: Container(

height: 180,

width: MediaQuery.of(context).size.width,

child: Column(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Text(

carmaker + ' ' + carmodel,

style: TextStyle(

color: Colors.white,

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Container(

height: 100,

width: 180,

decoration: BoxDecoration(

color: Colors.white.withAlpha(150),

borderRadius: BorderRadius.circular(12),

),

child: ClipRRect(

borderRadius: BorderRadius.circular(12),

child: Image.asset(

'assets/images/vehicles/' +

carPhotos[carmodel].toString(),

fit: BoxFit.cover,

),

),

),

Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

'Car Maker: ',

style: TextStyle(

fontSize: 13,

color: Colors.white,

),

),

Text(

carmaker,

style: TextStyle(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.white,

),

),

SizedBox(

height: 15,

),

Text(

'Car Model: ',

style: TextStyle(

fontSize: 13,

color: Colors.white,

),

),

Text(

carmodel,

style: TextStyle(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.white,

),

),

],

),

],

),

],

),

),

),

],

),

),

);

}

}

**carmodel.dart**

class CarModel {

String? carmaker;

String? carmodel;

CarModel({

this.carmaker,

this.carmodel,

});

}

**checkscreen.dart**

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

class checkscreen extends StatefulWidget {

const checkscreen({Key? key}) : super(key: key);

@override

State<checkscreen> createState() => \_checkscreenState();

}

class \_checkscreenState extends State<checkscreen> {

UserServices userServices = UserServices();

@override

Widget build(BuildContext context) {

return Scaffold(

body: Center(

child: ElevatedButton(

onPressed: () {},

child: Text("Click me"),

),

),

);

}

}

**commonvars.dart**

List months = [

'',

'January',

'February',

'March',

'April',

'May',

'June',

'July',

'August',

'September',

'October',

'November',

'December'

];

List days = [

'Sun',

'Mon',

'Tue',

'Wed',

'Thu',

'Fri',

'Sat',

'Sun',

];

**constants.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

FirebaseAuth firebaseAuth = FirebaseAuth.instance;

FirebaseFirestore firestore = FirebaseFirestore.instance;

CollectionReference userbase = firestore.collection('userdata');

String userkey = '';

String username = '';

**dashboard.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/homeparent.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleshop.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/carcarousal.dart';

import 'package:vehicle\_maintenance\_app/widgets/loadingblock.dart';

class Dashboard extends StatefulWidget {

final GlobalKey<ScaffoldState> mykey;

const Dashboard({Key? key, required this.mykey}) : super(key: key);

@override

State<Dashboard> createState() => \_DashboardState();

}

class \_DashboardState extends State<Dashboard> {

UserServices userServices = UserServices();

List<String> carkeys = [];

int currentpage = 0;

bool refreshbottom = false;

void setCurrentpage(int page) {

setState(() {

currentpage = page;

});

print(currentpage.toString() + 'from dashboard');

print(carkeys.length.toString());

print(carkeys);

}

Future<QuerySnapshot> getcardata() async {

QuerySnapshot data = await userServices.getcars();

carkeys.clear();

for (DocumentSnapshot snapshot in data.docs) {

carkeys.add(snapshot.id.toString());

}

if (refreshbottom == false) {

// to set the option visible

refreshbottom = true;

setState(() {});

}

return data;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

automaticallyImplyLeading: false,

backgroundColor: Colors.white,

title: Text(

'Dashboard',

style: TextStyle(

fontWeight: FontWeight.bold,

),

),

actions: [

IconButton(

onPressed: () {

widget.mykey.currentState?.openEndDrawer();

},

icon: Icon(

Icons.menu,

),

),

],

),

body: Container(

child: Column(

children: [

Container(

margin: EdgeInsets.symmetric(horizontal: globalpadding),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

'Shops',

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

),

),

SizedBox(

height: 10,

),

Container(

// height: 70,

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

shopcircle(imgname: 'lg1.jpg'),

shopcircle(imgname: 'lg2.jpg'),

shopcircle(imgname: 'lg3.jpg'),

shopcircle(imgname: 'lg4.jpg'),

SizedBox(

width: 20,

),

Icon(Icons.search),

],

),

),

//shop circles

SizedBox(

height: 15,

),

],

),

),

Expanded(

child: FutureBuilder(

future: getcardata(),

builder: (context, snapshot) {

if (snapshot.hasData && snapshot.data!.size != 0) {

return Column(

children: [

Container(

height: 180,

width: MediaQuery.of(context).size.width,

child: carCarousal(

setCurrentpage: setCurrentpage,

items: [

for (DocumentSnapshot documentsnapshot

in snapshot.data!.docs)

Stack(

children: [

buildVehiclecard(

carmaker:

documentsnapshot.get('carmaker'),

carmodel:

documentsnapshot.get('carmodel'),

),

Positioned(

top: 0,

right: 0,

child: IconButton(

icon: Icon(

Icons.delete\_forever\_rounded,

color: Colors.white,

),

onPressed: () async {

bool? result =

await getdeleteconfirmation(

context);

print(result);

if (result == true) {

loadingBlock(context: context);

await userServices.deleteCar(

carkey: (currentpage ==

carkeys.length)

? (carkeys[currentpage - 1])

: (carkeys[currentpage]),

);

Navigator.pop(context);

setState(() {});

}

},

),

)

],

),

addnewvehicle(),

],

),

),

Expanded(

child: Container(

child: Visibility(

visible: (currentpage == carkeys.length)

? (false)

: (true),

child: SingleChildScrollView(

physics: BouncingScrollPhysics(),

child: Column(

children: [

SizedBox(

height: 15,

),

paymentTile(

carkey: (currentpage == carkeys.length)

? (carkeys[currentpage - 1])

: (carkeys[currentpage]),

),

SizedBox(

height: 15,

),

appointmentTile(

carkey: (currentpage == carkeys.length)

? (carkeys[currentpage - 1])

: (carkeys[currentpage]),

),

SizedBox(

height: 15,

),

scheduleAppointmentTile(

carkey: (currentpage == carkeys.length)

? (carkeys[currentpage - 1])

: (carkeys[currentpage]),

),

SizedBox(

height: 20,

),

],

),

),

),

),

),

],

);

} else if (snapshot.hasData && snapshot.data!.size == 0) {

return Center(

child: FloatingActionButton.extended(

onPressed: () async {

await Navigator.pushNamed(context, '/addnewvehicle');

setState(() {});

},

backgroundColor: maintheme.withAlpha(200),

foregroundColor: Colors.white,

label: Text('Add New Vehicle'),

icon: Icon(Icons.add),

),

);

} else {

return Container(

height: 100,

width: 100,

child: FittedBox(

child: CircularProgressIndicator(

strokeWidth: 2,

),

),

);

}

},

),

),

],

),

),

);

}

getdeleteconfirmation(context) {

return showDialog(

context: context,

barrierDismissible: false,

builder: (context) {

return AlertDialog(

title: Text('Delete'),

content: Text('Do you want to delete this car?'),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

actions: [

TextButton(

onPressed: () {

Navigator.pop(context);

},

child: Text("Cancel"),

),

FilledButton(

onPressed: () {

Navigator.pop(context, true);

},

style: FilledButton.styleFrom(

backgroundColor: Colors.red,

),

child: Text('Delete'),

),

],

);

});

}

addnewvehicle() {

return Container(

height: 180,

width: MediaQuery.of(context).size.width - 30,

decoration: BoxDecoration(

color: maintheme.withAlpha(50),

borderRadius: BorderRadius.circular(16),

),

child: Center(

child: FloatingActionButton.extended(

onPressed: () async {

await Navigator.pushNamed(context, '/addnewvehicle');

setState(() {});

},

backgroundColor: maintheme.withAlpha(200),

foregroundColor: Colors.white,

label: Text('Add New Vehicle'),

icon: Icon(Icons.add),

),

),

);

}

Widget shopcircle({String imgname = 'lg1.jpg'}) {

return Container(

height: MediaQuery.of(context).size.width / 6,

width: MediaQuery.of(context).size.width / 6,

decoration: BoxDecoration(

shape: BoxShape.circle,

border: Border.all(

color: maintheme.withAlpha(50),

width: 5,

),

),

child: ClipRRect(

borderRadius: BorderRadius.circular(100),

child: Image.asset(

'assets/logos/' + imgname,

fit: BoxFit.fill,

),

),

);

}

}

class scheduleAppointmentTile extends StatelessWidget {

final String carkey;

const scheduleAppointmentTile({Key? key, required this.carkey})

: super(key: key);

@override

Widget build(BuildContext context) {

return Container(

padding: EdgeInsets.symmetric(horizontal: globalpadding),

child: ElevatedButton(

onPressed: () {

Navigator.pushNamed(context, '/scheduleshop',

arguments: ServiceModel(carkey: carkey));

},

style: ElevatedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

elevation: 7,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

),

child: Container(

height: 100,

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Row(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Stack(

children: [

iconMaker(

iconData: Icons.calendar\_month,

),

Positioned(

right: 3,

top: 3,

child: Container(

height: 7,

width: 7,

decoration: BoxDecoration(

color: Colors.red,

borderRadius: BorderRadius.circular(20)),

),

),

],

),

SizedBox(

width: 20,

),

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

"SCHEDULE APPOINTMENT",

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

color: Colors.black),

),

SizedBox(

height: 5,

),

Text(

"Time to give your car some love",

style: TextStyle(

fontSize: 12,

color: Colors.black,

),

),

],

),

),

IconButton(

onPressed: () {},

icon: Icon(

Icons.chevron\_right\_rounded,

size: 30,

color: Colors.black,

),

),

],

),

],

),

),

),

);

}

}

class iconMaker extends StatelessWidget {

final IconData iconData;

const iconMaker({Key? key, this.iconData = Icons.access\_time})

: super(key: key);

@override

Widget build(BuildContext context) {

return Container(

height: 45,

width: 45,

decoration: BoxDecoration(

color: darktext.withAlpha(30),

shape: BoxShape.circle,

),

child: Icon(

iconData,

color: darktext,

size: 30,

),

);

}

}

class appointmentTile extends StatelessWidget {

final String carkey;

const appointmentTile({Key? key, required this.carkey}) : super(key: key);

@override

Widget build(BuildContext context) {

return Container(

padding: EdgeInsets.symmetric(horizontal: globalpadding),

child: ElevatedButton(

onPressed: () {

Navigator.pushNamed(context, '/upcomingappointments',

arguments: [carkey, '']);

},

style: ElevatedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

elevation: 7,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

),

child: Container(

height: 100,

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Row(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Stack(

children: [

iconMaker(

iconData: Icons.timelapse\_rounded,

),

Positioned(

right: 3,

top: 3,

child: Container(

height: 7,

width: 7,

decoration: BoxDecoration(

color: Colors.red,

borderRadius: BorderRadius.circular(20)),

),

),

],

),

SizedBox(

width: 20,

),

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

"UPCOMING APPOINTMENTS",

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

color: Colors.black),

),

SizedBox(

height: 5,

),

Text(

"You Have an Appointment at MIDAS in 3 days",

style: TextStyle(

fontSize: 12,

color: Colors.black,

),

),

],

),

),

IconButton(

onPressed: () {},

icon: Icon(

Icons.chevron\_right\_rounded,

size: 30,

color: Colors.black,

),

),

],

),

],

),

),

),

);

}

}

class paymentTile extends StatefulWidget {

final String carkey;

const paymentTile({Key? key, required this.carkey}) : super(key: key);

@override

State<paymentTile> createState() => \_paymentTileState();

}

class \_paymentTileState extends State<paymentTile> {

UserServices userServices = UserServices();

bool expanded = false;

int totalcost = 0;

getsnapshotlist() async {

totalcost = 0;

print('made 0');

List<DocumentSnapshot> snapshots = [];

QuerySnapshot querySnapshot =

await userServices.getunpaidservicewithcarkey(carkey: widget.carkey);

for (DocumentSnapshot doc in querySnapshot.docs) {

snapshots.add(doc);

totalcost += int.parse(doc.get('serviceprice'));

}

return snapshots;

}

@override

Widget build(BuildContext context) {

return Container(

padding: EdgeInsets.symmetric(horizontal: globalpadding),

child: ElevatedButton(

onPressed: () {

setState(() {

expanded = !expanded;

});

},

style: ElevatedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

padding: EdgeInsets.all(0),

elevation: 7,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

),

child: AnimatedContainer(

duration: Duration(milliseconds: 500),

curve: Curves.fastLinearToSlowEaseIn,

width: MediaQuery.of(context).size.width,

padding: EdgeInsets.symmetric(horizontal: 15, vertical: 0),

height: (expanded) ? (180) : (100),

child: SingleChildScrollView(

physics: NeverScrollableScrollPhysics(),

scrollDirection: Axis.vertical,

child: FutureBuilder(

future: getsnapshotlist(),

builder: (context, snapshot) {

if (snapshot.hasData) {

return Column(

mainAxisAlignment: MainAxisAlignment.start,

children: [

Container(

height: 100,

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Row(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Stack(

children: [

iconMaker(

iconData: Icons.notifications\_rounded,

),

Positioned(

right: 3,

top: 3,

child: Container(

height: 7,

width: 7,

decoration: BoxDecoration(

color: (totalcost > 0)

? (Colors.red)

: (Colors.transparent),

borderRadius:

BorderRadius.circular(20)),

),

),

],

),

SizedBox(

width: 20,

),

Expanded(

child: Column(

crossAxisAlignment:

CrossAxisAlignment.start,

mainAxisAlignment:

MainAxisAlignment.center,

children: [

Text(

"PAYMENTS DUE",

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

color: (expanded)

? (Colors.black.withAlpha(100))

: (Colors.black),

),

),

SizedBox(

height: 5,

),

(expanded)

? (Text(

'Total: ₹ ' +

totalcost.toString(),

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

color: Colors.black,

),

))

: (Text(

(totalcost > 0)

? ("You have payments due")

: ("You have no payments due"),

style: TextStyle(

fontSize: 12,

color: Colors.black,

),

)),

],

),

),

SizedBox(

width: 20,

),

IconButton(

onPressed: () {

setState(() {

expanded = !expanded;

});

},

icon: AnimatedRotation(

duration: Duration(milliseconds: 500),

curve: Curves.fastLinearToSlowEaseIn,

turns: ((expanded) ? (1) : (0)) / 4,

child: Icon(

Icons.chevron\_right\_rounded,

size: 30,

color: Colors.black,

),

),

),

],

),

],

),

),

ClipRect(

child: (expanded)

? (Row(

children: [

Expanded(

flex: 4,

child: Text(

'Oil Change, standard checkup at Luffy Lube',

style: TextStyle(

fontSize: 12,

fontWeight: FontWeight.bold,

color: Colors.black,

),

),

),

SizedBox(

width: 10,

),

Expanded(

flex: 2,

child: OutlinedButton(

onPressed: () {

Navigator.pushNamed(

context, '/upcomingappointments',

arguments: [

widget.carkey,

'Payment Dues'

]);

},

style: OutlinedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: Colors.red,

shape: RoundedRectangleBorder(

borderRadius:

BorderRadius.circular(12),

),

side: BorderSide(

color: Colors.red,

width: 1,

),

),

child: Text(

'Review',

style: TextStyle(

color: Colors.red,

fontSize: 15,

),

),

),

),

],

))

: (Container()),

),

],

);

} else {

return Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Row(),

LinearProgressIndicator(),

],

);

}

},

)),

),

),

);

}

}

**data.dart**

import 'package:vehicle\_maintenance\_app/models/shop\_model.dart';

Map<String, dynamic> carMakers = {

'Mahindra': [

'XUV300',

'XUV500',

'XUV700',

'Thar',

],

'Tata': [

'Harrier',

'Nexon',

'Hexa',

'Punch',

],

'Hyundai': [

'Creta',

'Xcent',

'Venue',

'Verna',

],

'Honda': [

'Amaze',

'City',

'Accord',

'Civic',

],

};

Map<String, String> carPhotos = {

'XUV300': 'xuv300.png',

'XUV500': 'xuv500.png',

'XUV700': 'xuv700.png',

'Thar': 'thar.webp',

'Harrier': 'harrier.png',

'Nexon': 'nexon.png',

'Hexa': 'hexa.png',

'Punch': 'punch.jpeg',

'Creta': 'creta.png',

'Xcent': 'xcent.png',

'Venue': 'venue.png',

'Verna': 'verna.png',

'Amaze': 'amaze.png',

'City': 'city.png',

'Accord': 'accord.png',

'Civic': 'civic.png',

};

Map<String, int> services = {

'General check-up and inspection': 4000,

'Engine oil change': 1770,

'Air filter cleaning or replacement': 2300,

'Brake system inspection and repair': 3100,

'Wheel alignment and balancing': 1600,

'Battery check and replacement': 2900,

'Suspension system inspection and repair': 3500,

'Transmission system service': 2850,

'Fuel system cleaning': 2700,

'Spark plug replacement': 1040,

'Coolant system flush and refill': 1500,

'Timing belt replacement': 3000,

'Power steering system service': 2100,

'AC system service and repair': 4020,

'Exhaust system inspection and repair': 2700,

};

List<ShopModel> shopdata = [

ShopModel(

shopname: 'A to Z Motor Cycle Service Center',

shopaddress:

'R26C+PQX, Bajanai kovil Main Rd, Kavanur R.F.R[31]C, Tamil Nadu 603203',

shopphone: '+91 9892327504',

),

ShopModel(

shopname: 'S-Drive Multibrand Car Service - Perumbakkam',

shopaddress:

'No 8, 202c, Nookampalayam Rd, Perumbakkam, Chennai, Tamil Nadu 600126',

shopphone: '+91 9196937586',

),

ShopModel(

shopname: 'Vijay Automobiles Kattankulathur (car and bike)',

shopaddress:

'No 20 , Humming bird street, near Vgn Southern Avenue, apts, Kattankulathur, Tamil Nadu 603203',

shopphone: '+91 9395562173',

),

ShopModel(

shopname: 'Yamaha Service Centre (Bikes)',

shopaddress:

'No.50, NH-1, Vallal MGR Salai, Opp. Railway Station, Maraimalai Nagar, Tamil Nadu 603209',

shopphone: '+91 95660 09898',

),

ShopModel(

shopname: 'Maruti Suzuki Service (Vishnu Cars)',

shopaddress:

'No 19, GST Rd Potheri, Kattankulathur, Guduvancheri, Tamil Nadu 603202',

shopphone: '044 6620 5616',

),

ShopModel(

shopname: 'MG Automobile ( Car and Bike)',

shopaddress: 'Pillayar Koil St, Potheri, Kattankulathur, Tamil Nadu 603203',

shopphone: '+91 9854056414',

),

ShopModel(

shopname: 'MAHARAJA AUTO MOBILE & GARAGE (Car and Bike)',

shopaddress:

'Thailavaram Bus Stop, Thailavaram Village R2JW+9MG Chennai - Theni Highway, Chennai - Theni Hwy, Potheri, Kattankulathur, TamilNadu 603202',

shopphone: '+91 9618683380',

),

];

Map finaldata = {

// sample

'name': 'dhanush',

'cars': {

'key': {

'carmaker': 'carmaker',

'carmodel': 'carmodel',

},

},

'services': {

'servicekey': {

'carkey': 'carkey',

'shopname': 'shopname',

'servicename': 'servicename',

'serviceprice': 'serviceprice',

'servicedate': 'servicedate',

'servicetime': 'servicetime',

'notes': 'notes of the service',

'paymentstatus': 'paymentstatus', // completed, pending

},

},

};

**decision.dart**

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

logindecision() async {

SharedPreferences sharedpref = await SharedPreferences.getInstance();

String? spuserkey = sharedpref.getString('userkey');

String? spusername = sharedpref.getString('username');

if (spuserkey != null &&

spusername != null &&

spuserkey != '' &&

spusername != '') {

print(spuserkey);

userkey = spuserkey;

username = spusername;

return '/home';

} else {

return '/login';

}

}

**global.dart**

import 'package:flutter/material.dart';

double globalpadding = 15;

const Color maintheme = Color(0xff5658D6);

const Color darktext = Color(0xff090446);

TextStyle subtitle = TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

color: darktext,

);

**historyscreen.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/servicetile.dart';

class historyScreen extends StatefulWidget {

const historyScreen({Key? key}) : super(key: key);

@override

State<historyScreen> createState() => \_historyScreenState();

}

class \_historyScreenState extends State<historyScreen> {

Map<String, List<DocumentSnapshot>> carservices = {};

final UserServices userServices = UserServices();

Future<Map<String, List<DocumentSnapshot>>> getdata() async {

QuerySnapshot querySnapshot = await userServices.getserviceswithuserkey();

carservices.clear();

for (DocumentSnapshot documentSnapshot in querySnapshot.docs) {

carservices.putIfAbsent(documentSnapshot.get('carkey'), () => []);

carservices[documentSnapshot.get('carkey').toString()]

?.add(documentSnapshot);

}

return carservices;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

automaticallyImplyLeading: false,

backgroundColor: Colors.white,

title: Text(

'History',

style: TextStyle(

fontWeight: FontWeight.bold,

),

),

),

body: Container(

child: FutureBuilder(

future: getdata(),

builder: (context, snapshot) {

if (snapshot.hasData && snapshot.data!.length != 0) {

return ListView.builder(

padding: EdgeInsets.symmetric(vertical: 15),

physics: BouncingScrollPhysics(),

itemCount: snapshot.data!.length,

itemBuilder: (context, i) {

return section(

carkey: snapshot.data!.keys.elementAt(i),

documentSnapshot: snapshot.data!.values.elementAt(i),

);

},

);

} else if (snapshot.hasData && snapshot.data!.length == 0) {

return Center(

child: Text(

'No Service History Found',

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

color: darktext,

),

),

);

} else

return Center(child: CircularProgressIndicator());

},

),

),

);

}

Widget section(

{required String carkey,

required List<DocumentSnapshot> documentSnapshot}) {

return Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 5,

),

Container(

padding: EdgeInsets.symmetric(horizontal: 15, vertical: 10),

child: FutureBuilder(

future:

userbase.doc(userkey).collection('cars').doc(carkey).get(),

builder: (context, snapshot) {

if (snapshot.hasData) {

return Text(

snapshot.data!.get('carmaker') +

' ' +

snapshot.data!.get('carmodel') +

": ",

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

color: darktext,

),

);

} else

return Container(

padding: EdgeInsets.symmetric(vertical: 10),

width: 60,

child: LinearProgressIndicator(),

);

}),

),

Column(

children: [

for (DocumentSnapshot ds in documentSnapshot)

serviceTile(documentSnapshot: ds),

],

),

],

);

}

}

**homescreen.dart**

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/dashboard.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/historyscreen.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/schedulescreen.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/billingmain.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

import 'package:vehicle\_maintenance\_app/widgets/loadingblock.dart';

class homeParent extends StatefulWidget {

const homeParent({Key? key}) : super(key: key);

@override

State<homeParent> createState() => \_homeParentState();

}

class \_homeParentState extends State<homeParent> {

late PageController pageController;

int bottomindex = 2;

double globalpadding = 15;

TextStyle dummystyle = TextStyle(

fontSize: 25,

fontWeight: FontWeight.bold,

color: darktext,

);

GlobalKey<ScaffoldState> \_globalKey = GlobalKey<ScaffoldState>();

void changePage(int pageindex) {

setState(() {

bottomindex = pageindex;

pageController.animateToPage(pageindex,

duration: Duration(milliseconds: 500),

curve: Curves.fastLinearToSlowEaseIn);

});

}

@override

void initState() {

// TODO: implement initState

super.initState();

pageController = PageController(initialPage: bottomindex);

}

@override

Widget build(BuildContext context) {

return Scaffold(

key: \_globalKey,

// appBar: AppBar(),

body: PageView(

controller: pageController,

physics: BouncingScrollPhysics(),

onPageChanged: (i) {

setState(() {

bottomindex = i;

});

},

children: [

billingMain(),

// Container(

// alignment: Alignment.center,

// child: Text(

// 'Schedule',

// style: dummystyle,

// ),

// ),

scheduleScreen(),

Dashboard(mykey: \_globalKey),

historyScreen(),

// Container(

// alignment: Alignment.center,

// child: Text(

// 'History',

// style: dummystyle,

// ),

// ),

Container(

alignment: Alignment.center,

child: Text(

'Messages',

style: dummystyle,

),

),

],

),

endDrawer: Drawer(

backgroundColor: maintheme,

width: 300,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.only(

topLeft: Radius.circular(22),

bottomLeft: Radius.circular(22),

),

),

child: Container(

child: Column(

children: [

SizedBox(

height: 100,

),

Container(

margin: EdgeInsets.only(left: 25),

child: Row(

children: [

Card(

elevation: 7,

shape: CircleBorder(),

child: CircleAvatar(

radius: 35,

child: ClipRRect(

borderRadius: BorderRadius.circular(200),

child: Image.asset('assets/images/profilephoto.png'),

),

backgroundColor: Colors.black,

),

),

SizedBox(

width: 15,

),

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

'Welcome',

style: TextStyle(

fontWeight: FontWeight.bold,

fontSize: 20,

color: Colors.white,

),

),

SizedBox(

height: 5,

),

Text(

username,

maxLines: 1,

overflow: TextOverflow.ellipsis,

style: TextStyle(

fontSize: 18,

color: Colors.white,

),

),

],

),

),

SizedBox(

width: 15,

),

],

),

),

SizedBox(

height: 35,

),

Column(

children: [

draweritem(

title: 'Vehicle',

icon: Icons.car\_rental\_rounded,

),

draweritem(

title: 'Settings',

icon: Icons.settings\_rounded,

),

draweritem(

title: 'Help',

icon: Icons.help\_outline\_rounded,

),

draweritem(

title: 'About',

icon: Icons.info\_outline\_rounded,

),

draweritem(

title: 'Invite',

icon: Icons.insert\_invitation\_rounded,

),

draweritem(

title: 'Log Out',

icon: Icons.logout\_rounded,

onPressed: () async {

loadingBlock(context: context);

await firebaseAuth.signOut();

SharedPreferences sharedpref =

await SharedPreferences.getInstance();

sharedpref.setString('userkey', '');

sharedpref.setString('username', '');

Navigator.pop(context);

Navigator.pushReplacementNamed(context, '/login');

},

),

],

),

Spacer(),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'VehiCare',

style: TextStyle(

fontWeight: FontWeight.bold,

fontSize: 30,

color: Colors.white,

),

),

SizedBox(

width: 20,

),

// Icon(

// Icons.car\_crash\_rounded,

// color: Colors.red,

// size: 60,

// ),

Container(

height: 60,

width: 60,

decoration: BoxDecoration(

boxShadow: [

BoxShadow(

blurRadius: 20,

color: Colors.black.withAlpha(50),

),

],

),

child: Image.asset(

'assets/images/logo\_black.png',

color: Colors.white,

),

),

],

),

SizedBox(

height: 30,

),

],

),

),

),

bottomNavigationBar: BottomNavigationBar(

backgroundColor: maintheme,

type: BottomNavigationBarType.fixed,

currentIndex: bottomindex,

selectedFontSize: 12,

unselectedFontSize: 10,

iconSize: 25,

selectedItemColor: Colors.white,

unselectedItemColor: Colors.white.withAlpha(150),

onTap: (index) {

setState(() {

changePage(index);

});

},

items: [

BottomNavigationBarItem(

icon: Icon(Icons.currency\_rupee\_rounded),

label: 'Billing',

),

BottomNavigationBarItem(

icon: Icon(Icons.schedule\_rounded),

label: 'Schedule',

),

BottomNavigationBarItem(

icon: Icon(Icons.dashboard\_customize\_rounded),

label: 'Dashboard',

),

BottomNavigationBarItem(

icon: Icon(Icons.history\_edu\_rounded),

label: 'History',

),

BottomNavigationBarItem(

icon: Icon(Icons.message\_rounded),

label: 'Messages',

),

],

),

);

}

Widget draweritem(

{title = 'title', required IconData icon, Function? onPressed}) {

return TextButton(

onPressed: () {

if (onPressed != null) {

onPressed();

}

},

style: TextButton.styleFrom(

foregroundColor: Colors.white,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

),

child: Container(

margin: EdgeInsets.only(top: 8, bottom: 8, left: 35),

child: Row(

children: [

Icon(

icon,

size: 20,

color: Colors.white,

),

SizedBox(

width: 15,

),

Text(

title,

style: TextStyle(

fontSize: 18,

color: Colors.white,

),

),

],

),

),

);

}

}

**loadingblock.dart**

import 'package:flutter/material.dart';

loadingBlock({required BuildContext context, bool exitable = false}) {

showDialog(

context: context,

barrierDismissible: false,

builder: (context) => WillPopScope(

onWillPop: () async {

return exitable;

},

child: AlertDialog(

contentPadding: EdgeInsets.symmetric(vertical: 30),

content: Container(

height: 50,

width: 50,

child: FittedBox(

child: CircularProgressIndicator(

strokeWidth: 3,

),

),

),

),

),

);

}

**loginpage.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/homeparent.dart';

import 'package:vehicle\_maintenance\_app/screens/signup.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

import 'package:vehicle\_maintenance\_app/services/user\_auth.dart';

class LoginPage extends StatefulWidget {

const LoginPage({Key? key}) : super(key: key);

@override

State<LoginPage> createState() => \_LoginPageState();

}

class \_LoginPageState extends State<LoginPage> {

TextEditingController emailcontroller = TextEditingController();

TextEditingController passwordcontroller = TextEditingController();

Authentication \_authentication = Authentication();

String errortext = '';

bool obscurepassword = true;

bool loading = false;

bool remember = true;

login() async {

String email, password;

setState(() {

loading = true;

});

email = emailcontroller.text;

password = passwordcontroller.text;

if (email != '' && password != '') {

var result = await \_authentication.loginUser(email, password);

if (result.runtimeType == UserCredential) {

Navigator.pushReplacementNamed(context, '/home');

} else if (result.toString().contains('invalid-email')) {

setState(() {

errortext = 'Invalid email Format';

});

} else if (result.toString().contains('user-not-found')) {

showDialog(

context: context,

builder: (context) => AlertDialog(

contentPadding: EdgeInsets.all(50),

title: Text(

'User not Found',

style: TextStyle(

fontWeight: FontWeight.bold,

fontSize: 20,

),

),

content: Text(

"Do you Want to SIGNUP ??",

style: TextStyle(

fontSize: 18,

),

),

actions: [

TextButton(

onPressed: () {

Navigator.pop(context);

},

style: TextButton.styleFrom(),

child: Text(

"Cancel",

style: TextStyle(),

),

),

SizedBox(

width: 10,

),

OutlinedButton(

onPressed: () {

Navigator.pushReplacementNamed(context, '/signup');

},

style: TextButton.styleFrom(

backgroundColor: maintheme,

foregroundColor: Colors.white,

),

child: Text(

"Sign Up",

style: TextStyle(

color: Colors.white,

fontWeight: FontWeight.bold,

),

),

),

],

),

);

} else if (result.toString().contains('wrong-password')) {

setState(() {

errortext = 'Oops, Wrong Password!!';

});

} else {

setState(() {

errortext = 'Login Successfully Failed';

});

}

}

setState(() {

loading = false;

});

}

bool rem = true;

@override

Widget build(BuildContext context) {

return Scaffold(

body: SafeArea(

child: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

children: [

Row(

mainAxisAlignment: MainAxisAlignment.end,

children: [

TextButton(

onPressed: () {

Navigator.pushReplacementNamed(context, '/signup');

},

child: Text(

"Sign Up",

style: TextStyle(

fontWeight: FontWeight.bold,

),

),

),

],

),

Expanded(

child: SingleChildScrollView(

physics: BouncingScrollPhysics(),

child: Container(

// height: MediaQuery.of(context).size.height,

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.center,

children: [

SizedBox(

height: 50,

),

Row(),

// Container(

// child: Icon(

// Icons.car\_crash\_sharp,

// color: Colors.redAccent,

// size: 100,

// ),

// ),

Container(

height: 150,

width: 150,

child: Image.asset(

'assets/images/logo\_color.png',

),

),

SizedBox(

height: 10,

),

Text(

"VehiCare",

style: TextStyle(

color: maintheme,

fontWeight: FontWeight.bold,

fontSize: 35,

),

),

SizedBox(

height: 30,

),

Container(

height: 60,

child: TextField(

controller: emailcontroller,

style: TextStyle(

fontSize: 18,

),

decoration: InputDecoration(

hintText: 'Email',

hintStyle: TextStyle(

color: Colors.grey,

fontSize: 18,

),

focusedBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 1,

),

),

enabledBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 0.5,

),

),

),

),

),

SizedBox(

height: 20,

),

Row(

children: [

Expanded(

child: Container(

height: 60,

child: TextField(

controller: passwordcontroller,

style: TextStyle(

fontSize: 18,

),

obscureText: obscurepassword,

obscuringCharacter: '\*',

decoration: InputDecoration(

hintText: 'Password',

hintStyle: TextStyle(

color: Colors.grey,

fontSize: 18,

),

focusedBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 1,

),

),

enabledBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 0.5,

),

),

),

),

),

),

SizedBox(

width: 5,

),

IconButton(

onPressed: () {

setState(() {

obscurepassword = !obscurepassword;

});

},

icon: (obscurepassword)

? (Icon(Icons.visibility\_outlined))

: (Icon(Icons.visibility\_off\_outlined)),

),

],

),

SizedBox(

height: 5,

),

Row(

children: [

TextButton(

onPressed: () {

setState(() {

remember =

(remember == true) ? (false) : (true);

});

},

style: TextButton.styleFrom(

padding: EdgeInsets.only(right: 15),

),

child: Row(

children: [

Checkbox(

value: remember,

shape: CircleBorder(),

activeColor: maintheme,

checkColor: Colors.white,

onChanged: (res) {},

),

Text(

'Remember Me',

style: TextStyle(

color: Colors.black,

fontSize: 12,

),

),

],

),

),

Spacer(),

TextButton(

onPressed: () {},

child: Text(

'Forgot Password?',

style: TextStyle(

color: maintheme,

fontSize: 14,

),

),

),

],

),

SizedBox(

height: 20,

),

TextButton(

onPressed: (loading == true)

? (null)

: (() {

login();

}),

style: TextButton.styleFrom(

backgroundColor: maintheme,

foregroundColor: Colors.white,

padding: EdgeInsets.zero,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

),

child: Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.center,

children: [

(loading == true)

? (CircularProgressIndicator(

color: Colors.white,

))

: (Container(

height: 60,

alignment: Alignment.center,

child: Text(

"Log in",

style: TextStyle(

color: Colors.white,

fontSize: 20,

),

),

)),

],

),

),

SizedBox(

height: 50,

),

Text(

errortext,

style: TextStyle(

color: Colors.red,

fontWeight: FontWeight.bold,

),

),

],

),

),

),

),

],

),

),

),

);

}

}

**ongenerateroute.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/screens/addnewcar.dart';

import 'package:vehicle\_maintenance\_app/screens/loginpage.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/homeparent.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/paymentscreen.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleappointment.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleconfirmation.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/schedulereview.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleshop.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/schedulesuccess.dart';

import 'package:vehicle\_maintenance\_app/screens/signup.dart';

import 'package:vehicle\_maintenance\_app/screens/upcoming\_appointments.dart';

class RouteGenerator {

static Route generateRoute(RouteSettings settings) {

final args = settings.arguments;

switch (settings.name) {

// main screens

case '/login':

return MaterialPageRoute(builder: (\_) => LoginPage());

case '/signup':

return MaterialPageRoute(builder: (\_) => SignUp());

case '/home':

return MaterialPageRoute(builder: (\_) => homeParent());

//anonymous

case '/addnewvehicle':

return MaterialPageRoute(builder: (\_) => addNewCar());

case '/upcomingappointments':

if (args is List) {

return MaterialPageRoute(

builder: (\_) => upcomingAppointmentScreen(

carkey: args[0],

title: args[1],

));

} else

return errorroute();

//scheduling routes

case '/scheduleshop':

if (args is ServiceModel) {

return MaterialPageRoute(

builder: (\_) => scheduleShop(serviceModel: args));

} else

return errorroute();

case '/schedulereview':

if (args is ServiceModel)

return MaterialPageRoute(

builder: (\_) => scheduleReview(serviceModel: args));

else

return errorroute();

case '/scheduleappointment':

if (args is ServiceModel)

return MaterialPageRoute(

builder: (\_) => scheduleAppointment(serviceModel: args));

else

return errorroute();

case '/scheduleconfirmation':

if (args is ServiceModel)

return MaterialPageRoute(

builder: (\_) => scheduleConfirmation(serviceModel: args));

else

return errorroute();

case '/schedulesuccess':

return MaterialPageRoute(builder: (\_) => scheduleSuccess());

//payment screen routes

case '/paymentscreen':

if (args is List<DocumentSnapshot>)

return MaterialPageRoute(

builder: (\_) => paymentScreen(

servicesnapshots: args,

));

else

return errorroute();

// default

default:

return errorroute();

}

}

static Route errorroute() {

return MaterialPageRoute(builder: (\_) {

return Scaffold(

appBar: AppBar(

title: Text("Error Page"),

),

);

});

}

}

// schedule order

// 1. schedule shop

// 2. schedule review

// 3. schedule appointment

// 4. schedule confirmation

// 5. schedule success

**payment\_services.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

class PaymentServices {

payforService(List<DocumentSnapshot> snapshots) async {

for (DocumentSnapshot doc in snapshots) {

await userbase.doc(userkey).collection('services').doc(doc.id).update({

'paymentstatus': 'completed',

});

}

}

}

**paymentscreen.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/screens/payment/paymentsuccessfull.dart';

import 'package:vehicle\_maintenance\_app/services/payment\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/carcarousal.dart';

import 'package:vehicle\_maintenance\_app/widgets/loadingblock.dart';

TextStyle cardhead = TextStyle(

fontSize: 13,

color: Colors.white,

);

TextStyle cardans = TextStyle(

fontSize: 15,

color: Colors.white,

);

class paymentScreen extends StatefulWidget {

final List<DocumentSnapshot> servicesnapshots;

const paymentScreen({Key? key, required this.servicesnapshots})

: super(key: key);

@override

State<paymentScreen> createState() => \_paymentScreenState();

}

class \_paymentScreenState extends State<paymentScreen> {

int totalprice = 0;

PaymentServices paymentServices = PaymentServices();

payservices() async {

await paymentServices.payforService(widget.servicesnapshots);

print('completed');

}

getConfirmation(BuildContext context) {

showDialog(

context: context,

builder: (context) {

return CupertinoAlertDialog(

title: Text('Are you sure you want to complete payment?\n₹ ' +

totalprice.toString()),

actions: [

CupertinoDialogAction(

child: Text('Cancel'),

onPressed: () {

Navigator.pop(context);

},

),

CupertinoDialogAction(

child: Text('Yes'),

onPressed: () async {

loadingBlock(context: context);

await payservices();

Navigator.pop(context); // to pop loading block

Navigator.pop(context); // to pop dialog box

Navigator.pushNamedAndRemoveUntil(

context,

'/home',

(route) => false,

);

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => paymentSuccessfullScreen()));

},

),

],

);

},

);

}

calculatetotal() {

for (DocumentSnapshot doc in widget.servicesnapshots) {

totalprice += int.parse(doc.get('serviceprice'));

}

}

@override

void initState() {

// TODO: implement initState

super.initState();

calculatetotal();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

automaticallyImplyLeading: false,

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: TextButton(

onPressed: () {

Navigator.pop(context);

},

child: Text(

'Cancel',

style: TextStyle(

fontWeight: FontWeight.bold,

fontSize: 15,

),

),

),

leadingWidth: 100,

title: Text(

"Payment",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

// padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

children: [

SizedBox(

height: 15,

),

Row(

children: [

SizedBox(

width: 20,

),

Text(

'LINKED CARDS',

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

),

),

Spacer(),

Text(

'ADD NEW',

style: TextStyle(

fontSize: 14,

color: maintheme,

fontWeight: FontWeight.bold,

),

),

SizedBox(

width: 20,

),

],

),

SizedBox(

height: 10,

),

Container(

height: 260,

child: carCarousal(

items: [

buildCreditCard(),

buildCreditCard(),

buildCreditCard(),

],

),

),

SizedBox(

height: 15,

),

Expanded(

child: Padding(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

children: [

Expanded(

child: Card(

elevation: 7,

surfaceTintColor: Colors.white,

color: Colors.white,

child: Container(

padding: EdgeInsets.all(25),

width: MediaQuery.of(context).size.width,

child: Column(

children: [

Row(

mainAxisAlignment:

MainAxisAlignment.spaceBetween,

children: [

Expanded(

child: Column(

crossAxisAlignment:

CrossAxisAlignment.start,

mainAxisAlignment:

MainAxisAlignment.start,

children: [

Text(

widget.servicesnapshots.first

.get('shopname'),

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

),

),

SizedBox(

height: 5,

),

Text(

widget.servicesnapshots.first

.get('servicedate'),

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

color: darktext.withAlpha(100),

),

),

],

),

),

IconButton(

onPressed: () {},

icon: Icon(

Icons.info\_outline\_rounded,

color: maintheme,

),

),

],

),

SizedBox(

height: 15,

),

Expanded(

child: ListView.builder(

itemCount: widget.servicesnapshots.length,

physics: BouncingScrollPhysics(),

itemBuilder: (context, i) {

return billdetails(

servicename: widget.servicesnapshots[i]

.get('servicename'),

price: widget.servicesnapshots[i]

.get('serviceprice'),

);

}),

),

SizedBox(

height: 10,

),

Row(

mainAxisAlignment: MainAxisAlignment.end,

children: [

Text(

'Total: ₹ ' + totalprice.toString(),

style: TextStyle(

color: maintheme,

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

],

),

SizedBox(

height: 10,

),

Row(

children: [

ClipRRect(

borderRadius: BorderRadius.circular(10),

child: Container(

height: 30,

width: 50,

child: Image.asset(

'assets/images/creditcardbg.jpg',

),

),

),

SizedBox(

width: 10,

),

Text(

'My Personal Card',

style: TextStyle(

color: Colors.black,

fontSize: 14,

fontWeight: FontWeight.bold,

),

),

],

)

],

),

),

),

),

SizedBox(

height: 15,

),

],

),

),

),

Padding(

padding: EdgeInsets.symmetric(vertical: 15, horizontal: 15),

child: OutlinedButton(

onPressed: () {

getConfirmation(context);

},

style: OutlinedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: Colors.red,

padding: EdgeInsets.symmetric(vertical: 17),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

side: BorderSide(

width: 1,

color: Colors.red,

),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Complete Payment',

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: Colors.red,

),

),

],

),

),

),

],

),

),

);

}

}

class billdetails extends StatelessWidget {

final String servicename;

final String price;

const billdetails({Key? key, this.servicename = 'Detail', this.price = '50'})

: super(key: key);

@override

Widget build(BuildContext context) {

return Container(

padding: EdgeInsets.symmetric(vertical: 2),

child: Row(

children: [

Expanded(

child: Text(

servicename,

maxLines: 1,

overflow: TextOverflow.ellipsis,

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

color: darktext,

),

),

),

SizedBox(

width: 15,

),

Text(

'₹ ' + price.toString(),

style: TextStyle(

fontSize: 14,

fontWeight: FontWeight.bold,

color: darktext,

),

),

],

),

);

}

}

class buildCreditCard extends StatelessWidget {

const buildCreditCard({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return ClipRRect(

borderRadius: BorderRadius.circular(22),

child: Container(

height: 240,

alignment: Alignment.center,

width: MediaQuery.of(context).size.width - 30,

child: Stack(

children: [

Container(

// height: 240,

width: MediaQuery.of(context).size.width,

child: Image.asset(

'assets/images/creditcardbg.jpg',

fit: BoxFit.fill,

),

),

Container(

// height: 240,

// width: MediaQuery.of(context).size.width,

padding: EdgeInsets.all(25),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.max,

children: [

Text(

'My Personal Card',

style: TextStyle(

fontSize: 18,

color: Colors.white,

fontWeight: FontWeight.bold,

),

),

SizedBox(

height: 25,

),

Expanded(

child: Row(

crossAxisAlignment: CrossAxisAlignment.stretch,

children: [

Expanded(

flex: 5,

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: [

Text(

'Credit Card Number',

style: cardhead,

),

Text(

'xxxx - xxxx - xxxx - 4783',

style: cardans,

),

],

),

Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: [

Text(

'Name on the Card',

style: cardhead,

),

Text(

'DREW FULLER',

style: cardans,

),

],

),

],

),

),

Expanded(

flex: 2,

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: [

Text(

'Expiration',

style: cardhead,

),

Text(

'09/23',

style: cardans,

),

],

),

Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: [

Text(

'CVV/CVS',

style: cardhead,

),

Text(

'\*\*\*',

style: cardans,

),

],

),

// Text(

// 'Credit Card Number',

// style: cardhead,

// ),

// Text(

// 'xxxx - xxxx - xxxx - 4783',

// style: cardans,

// ),

// Text(

// 'Credit Card Number',

// style: cardhead,

// ),

// Text(

// 'xxxx - xxxx - xxxx - 4783',

// style: cardans,

// ),

],

),

),

],

),

),

SizedBox(

height: 15,

),

],

),

),

Positioned(

bottom: 10,

right: 10,

child: IconButton(

onPressed: () {},

icon: Icon(

Icons.edit\_outlined,

color: Colors.white.withAlpha(150),

size: 20,

),

),

),

],

),

),

);

}

}

**paymentsuccessful.dart**

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

class paymentSuccessfullScreen extends StatelessWidget {

const paymentSuccessfullScreen({Key? key}) : super(key: key);

final String servicehead = "Successfully Paid";

final String servicedes = "Your Payment has been Successfully Completed";

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

automaticallyImplyLeading: false,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

title: Text(

"Confirmation",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

padding: EdgeInsets.all(35),

child: Column(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: [

Text(

servicehead,

textAlign: TextAlign.center,

style: TextStyle(

fontSize: 30,

fontWeight: FontWeight.bold,

color: darktext,

),

),

Text(

servicedes,

textAlign: TextAlign.center,

style: TextStyle(

fontSize: 18,

),

),

OutlinedButton(

onPressed: () {

Navigator.pop(context);

},

style: OutlinedButton.styleFrom(

side: BorderSide(

color: maintheme,

),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

),

child: Container(

padding: EdgeInsets.all(15),

child: Text(

'Back to Billing Menu',

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

),

),

),

),

Container(

height: 150,

width: 150,

child: Image.asset(

'assets/images/logo\_color.png',

),

),

SizedBox(

height: 60,

),

],

),

),

);

}

}

**paymenttiles.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

class recentTransactionsTile extends StatelessWidget {

const recentTransactionsTile({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Column(

children: [

TextButton(

onPressed: () {},

style: TextButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(10),

),

),

child: Container(

child: Row(

children: [

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 5,

),

Text(

'MIDAS',

style: TextStyle(

color: darktext,

fontSize: 14,

fontWeight: FontWeight.bold,

),

),

SizedBox(

height: 5,

),

Text(

'February 02, 2019',

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

Text(

'Tires, Brakes',

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

],

),

),

Row(

children: [

Text(

'\$ 125',

style: TextStyle(

fontSize: 20,

color: maintheme,

fontWeight: FontWeight.bold,

),

),

Icon(

Icons.chevron\_right\_rounded,

color: darktext,

size: 35,

),

],

),

],

),

),

),

Divider(

color: darktext.withAlpha(100),

),

],

);

}

}

class PaynowTile extends StatelessWidget {

final DocumentSnapshot snapshot;

const PaynowTile({Key? key, required this.snapshot}) : super(key: key);

@override

Widget build(BuildContext context) {

return Card(

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

color: Colors.white,

elevation: 7,

surfaceTintColor: Colors.white,

child: Container(

padding: EdgeInsets.all(15),

child: Row(

children: [

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

snapshot.get('shopname'),

maxLines: 1,

overflow: TextOverflow.ellipsis,

style: subtitle.copyWith(

fontSize: 15,

),

),

SizedBox(

height: 7,

),

Text(

snapshot.get('servicedate'),

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

SizedBox(

height: 4,

),

Text(

snapshot.get('servicename'),

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

],

),

),

SizedBox(

width: 15,

),

Column(

children: [

Text(

'₹ ' + snapshot.get('serviceprice').toString(),

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: maintheme,

),

),

SizedBox(

height: 10,

),

OutlinedButton(

onPressed: () {

Navigator.pushNamed(

context,

'/paymentscreen',

arguments: [snapshot],

);

},

style: OutlinedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: Colors.red,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

side: BorderSide(

color: Colors.red,

width: 1,

),

),

child: Text(

'Pay Now',

style: TextStyle(

color: Colors.red,

fontSize: 15,

),

),

),

],

),

],

),

),

);

}

}

**scheduleappointment.dart**

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/commonvars.dart';

import 'package:vehicle\_maintenance\_app/data.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleconfirmation.dart';

class scheduleAppointment extends StatefulWidget {

final ServiceModel serviceModel;

const scheduleAppointment({Key? key, required this.serviceModel})

: super(key: key);

@override

State<scheduleAppointment> createState() => \_scheduleAppointmentState();

}

class \_scheduleAppointmentState extends State<scheduleAppointment> {

int? servicevalue;

DateTime selecteddate = DateTime.now();

TimeOfDay selectedtime = TimeOfDay.fromDateTime(DateTime.now());

TextEditingController notes = TextEditingController();

@override

Widget build(BuildContext context) {

return Scaffold(

resizeToAvoidBottomInset: false,

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: IconButton(

onPressed: () {

Navigator.pop(context);

},

icon: Icon(

CupertinoIcons.left\_chevron,

color: darktext,

),

),

title: Text(

"Schedule",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

children: [

Expanded(

child: SingleChildScrollView(

physics: BouncingScrollPhysics(),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 15,

),

Text(

'SELECT SERVICE',

style: subtitle,

),

SizedBox(

height: 10,

),

Row(

children: [

Expanded(

child: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

decoration: BoxDecoration(

borderRadius: BorderRadius.circular(12),

border: Border.all(

width: 1,

color: maintheme,

),

),

child: DropdownButton(

items: [

for (int i = 0; i < services.length; i++)

DropdownMenuItem(

value: i,

child: Text(services.keys.elementAt(i)),

),

],

value: servicevalue,

alignment: AlignmentDirectional.centerStart,

borderRadius: BorderRadius.circular(12),

isExpanded: true,

style: TextStyle(

fontSize: 16,

color: darktext,

),

underline: Container(),

hint: Text('Services'),

icon: Icon(

CupertinoIcons.chevron\_down,

size: 20,

color: darktext,

),

onChanged: (a) {

setState(() {

servicevalue = a;

});

},

),

),

),

],

),

SizedBox(

height: 20,

),

Text(

'SELECT DATE AND TIME',

style: subtitle,

),

SizedBox(

height: 10,

),

Row(

children: [

Expanded(

flex: 5,

child: TextButton(

onPressed: () async {

DateTime? picked = await showDatePicker(

context: context,

initialDate: selecteddate,

firstDate: DateTime.now(),

lastDate: DateTime(2025, 2),

);

if (picked != null && picked != selecteddate) {

setState(() {

selecteddate = picked;

print(selecteddate);

});

}

},

style: TextButton.styleFrom(

side: BorderSide(

width: 1,

color: maintheme,

),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

),

child: Container(

padding: EdgeInsets.symmetric(vertical: 5),

child: Text(

selecteddate.day.toString() +

' / ' +

months[selecteddate.month] +

' / ' +

selecteddate.year.toString() +

' ( ' +

days[selecteddate.weekday] +

' )',

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

),

),

SizedBox(

width: 10,

),

Expanded(

flex: 3,

child: TextButton(

onPressed: () async {

TimeOfDay? pickedtime = await showTimePicker(

context: context,

initialTime:

TimeOfDay.fromDateTime(DateTime.now()),

);

if (pickedtime != null) {

setState(() {

selectedtime = pickedtime;

});

}

},

style: TextButton.styleFrom(

side: BorderSide(

width: 1,

color: maintheme,

),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

),

child: Container(

padding: EdgeInsets.symmetric(vertical: 5),

child: Text(

selectedtime.hourOfPeriod.toString() +

' : ' +

selectedtime.minute.toString() +

' ' +

selectedtime.period.name.toUpperCase(),

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

),

),

],

),

SizedBox(

height: 10,

),

Text(

"Note: The date and time will be confirmed by the service provider within 24 hours after scheduling through Vehicle Manager. You'll be notified by email, text or a phone call, based on your preference.",

textAlign: TextAlign.justify,

style: TextStyle(

fontSize: 13,

color: darktext.withAlpha(100),

),

),

SizedBox(

height: 20,

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Text(

'NOTES',

style: subtitle,

),

IconButton(

onPressed: () {},

icon: Icon(

Icons.info\_outline\_rounded,

color: maintheme,

),

)

],

),

SizedBox(

height: 10,

),

Container(

child: TextField(

controller: notes,

keyboardType: TextInputType.multiline,

minLines: 5,

maxLines: 5,

// expands: true,

decoration: InputDecoration(

hintText:

'Add additional notes about your visit and/or maintenance need',

border: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 1,

),

),

),

),

),

SizedBox(

height: 500,

),

],

),

),

),

TextButton(

onPressed: () {

if (servicevalue != null &&

selecteddate != null &&

selectedtime != null) {

print(servicevalue);

print(selecteddate.day);

print(selectedtime);

widget.serviceModel.servicename =

services.keys.elementAt(servicevalue!);

widget.serviceModel.serviceprice =

services.values.elementAt(servicevalue!).toString();

widget.serviceModel.day = selecteddate.day.toString();

widget.serviceModel.month = (selecteddate.month).toString();

widget.serviceModel.year = selecteddate.year.toString();

widget.serviceModel.hours = selectedtime.hour.toString();

widget.serviceModel.minutes = selectedtime.minute.toString();

widget.serviceModel.notes = notes.text;

Navigator.pushNamed(context, '/scheduleconfirmation',

arguments: widget.serviceModel);

}

widget.serviceModel.printer();

},

style: TextButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

padding: EdgeInsets.symmetric(vertical: 17),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

side: BorderSide(

width: 0.5,

color: darktext,

),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Continue',

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: darktext,

),

),

],

),

),

SizedBox(

height: 15,

),

],

),

),

);

}

}

**scheduleconfirmation.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/commonvars.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/dashboard.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/schedulesuccess.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/loadingblock.dart';

class scheduleConfirmation extends StatelessWidget {

final ServiceModel serviceModel;

scheduleConfirmation({Key? key, required this.serviceModel})

: super(key: key);

final UserServices userServices = UserServices();

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: IconButton(

onPressed: () {

Navigator.pop(context);

},

icon: Icon(

CupertinoIcons.left\_chevron,

color: darktext,

),

),

title: Text(

"Schedule",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 15,

),

Text(

'IS EVERYTHING CORRECT?',

style: subtitle,

),

SizedBox(

height: 10,

),

Card(

surfaceTintColor: Colors.white,

elevation: 7,

color: Colors.white,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

child: Container(

padding: EdgeInsets.all(15),

child: Column(

children: [

Text(

// "Oil Change at Jiffy Lube",

serviceModel.servicename! +

' at ' +

serviceModel.shopmodel!.shopname,

textAlign: TextAlign.center,

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: darktext,

),

),

SizedBox(

height: 20,

),

Row(

mainAxisSize: MainAxisSize.min,

mainAxisAlignment: MainAxisAlignment.start,

children: [

iconMaker(

iconData: CupertinoIcons.clock,

),

SizedBox(

width: 20,

),

Container(

width: 250,

child: Text(

// 'Monday, April 8 at 5:30 PM',

months[int.parse(serviceModel.month!)] +

' ' +

serviceModel.day +

' at ' +

serviceModel.hours +

':' +

serviceModel.minutes,

style: TextStyle(

fontSize: 13,

color: darktext.withAlpha(100),

fontWeight: FontWeight.bold,

),

),

),

],

),

SizedBox(

height: 20,

),

Row(

mainAxisSize: MainAxisSize.min,

mainAxisAlignment: MainAxisAlignment.start,

children: [

iconMaker(

iconData: Icons.location\_on\_outlined,

),

SizedBox(

width: 20,

),

Expanded(

child: Container(

child: Text(

// 'Jiffy Lube\n756, Barrington Road,\nHanover Park',

serviceModel.shopmodel!.shopname +

'\n' +

serviceModel.shopmodel!.shopaddress,

textAlign: TextAlign.start,

style: TextStyle(

fontSize: 13,

color: darktext.withAlpha(100),

fontWeight: FontWeight.bold,

),

),

),

),

],

),

],

),

),

),

SizedBox(

height: 20,

),

Text(

'CONTACT INFO',

style: subtitle,

),

SizedBox(

height: 10,

),

Card(

surfaceTintColor: Colors.white,

color: Colors.white,

elevation: 7,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

child: Container(

padding: EdgeInsets.all(25),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Text(

username,

style: TextStyle(

color: maintheme,

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

IconButton(

onPressed: () {},

icon: Icon(

Icons.edit\_outlined,

color: darktext,

size: 20,

),

),

],

),

SizedBox(

height: 10,

),

Text(

'+91 1234567890',

style: TextStyle(

color: maintheme,

fontSize: 14,

fontWeight: FontWeight.bold,

),

),

SizedBox(

height: 5,

),

Text(

'themailid@gmail.com',

style: TextStyle(

color: maintheme,

fontSize: 14,

fontWeight: FontWeight.bold,

),

),

],

),

),

),

Spacer(),

TextButton(

onPressed: () async {

loadingBlock(context: context);

await userServices.addSchedule(serviceModel: serviceModel);

Navigator.pop(context);

Navigator.pushNamedAndRemoveUntil(

context,

'/home',

ModalRoute.withName('/'),

);

Navigator.pushNamed(context, '/schedulesuccess');

// removing all from stack and

// pushing home under schedule succes screen

},

style: TextButton.styleFrom(

backgroundColor: darktext,

foregroundColor: Colors.white,

padding: EdgeInsets.symmetric(vertical: 17),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

side: BorderSide(

width: 0.5,

color: darktext,

),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Schedule appointment',

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: Colors.white,

),

),

],

),

),

SizedBox(

height: 15,

),

],

),

),

);

}

}

**schedulereview.dart**

import 'dart:developer';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/homeparent.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/scheduleappointment.dart';

class scheduleReview extends StatelessWidget {

final ServiceModel serviceModel;

const scheduleReview({Key? key, required this.serviceModel})

: super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: IconButton(

onPressed: () {

Navigator.pop(context);

},

icon: Icon(

CupertinoIcons.left\_chevron,

color: darktext,

),

),

title: Text(

"Schedule",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(

children: [

Card(

surfaceTintColor: Colors.white,

color: Colors.white,

elevation: 7,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

child: Container(

width: MediaQuery.of(context).size.width,

padding: EdgeInsets.all(20),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Expanded(

child: Text(

// 'Jiffy Lube',

serviceModel.shopmodel!.shopname.toString(),

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: darktext,

),

),

),

IconButton(

onPressed: () {},

icon: Icon(

Icons.favorite\_border\_rounded,

size: 30,

),

),

],

),

SizedBox(

height: 5,

),

ReviewStar(),

SizedBox(

height: 10,

),

Text(

// '756, Barrington Road, Hanover Park 5245',

serviceModel.shopmodel!.shopaddress,

style: subtitle.copyWith(

fontSize: 13,

color: darktext.withAlpha(100),

),

),

SizedBox(

height: 5,

),

linkfunc(

icon: Icons.call\_outlined,

text: 'Call ' +

serviceModel.shopmodel!.shopphone.toString(),

),

linkfunc(

icon: Icons.location\_on\_outlined,

text: 'Get Directions',

),

linkfunc(

icon: Icons.browse\_gallery\_outlined,

text: 'Go to Website',

),

SizedBox(

height: 15,

),

TextButton(

onPressed: () {

Navigator.pushNamed(context, '/scheduleappointment',

arguments: serviceModel);

},

style: TextButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

padding: EdgeInsets.symmetric(vertical: 17),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

side: BorderSide(

width: 0.5,

color: darktext,

),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Schedule Appoinement',

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

color: darktext,

),

),

],

),

),

],

),

),

),

SizedBox(

height: 5,

),

Row(

children: [

SizedBox(

width: 10,

),

Text(

"REVIEWS",

style: subtitle,

),

Spacer(),

IconButton(

onPressed: () {},

icon: Icon(

Icons.filter\_alt\_outlined,

size: 25,

color: darktext,

),

),

SizedBox(

width: 10,

),

],

),

SizedBox(

height: 0,

),

Expanded(

child: SingleChildScrollView(

physics: BouncingScrollPhysics(),

child: Column(

children: [

for (int i = 0; i < 3; i++) ReviewTile(),

],

),

),

),

TextButton(

onPressed: () {},

child: Text('See all Reviews'),

),

],

),

),

);

}

Widget linkfunc({required IconData icon, required String text}) {

return TextButton(

onPressed: () {},

style: TextButton.styleFrom(

padding: EdgeInsets.symmetric(

vertical: 0,

horizontal: 10,

),

),

child: Row(

children: [

Icon(

icon,

color: maintheme,

size: 20,

),

SizedBox(

width: 10,

),

Text(

text,

style: TextStyle(

color: darktext,

fontSize: 14,

fontWeight: FontWeight.bold,

),

)

],

),

);

}

}

class ReviewTile extends StatelessWidget {

const ReviewTile({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Column(

children: [

TextButton(

onPressed: () {},

style: TextButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: maintheme,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(10),

),

),

child: Container(

child: Row(

children: [

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Row(

children: [

ReviewStar(

a: 2,

),

SizedBox(

width: 10,

),

Text(

'1 Week ago',

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

],

),

SizedBox(

height: 5,

),

Text(

'by John Butler',

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

SizedBox(

height: 5,

),

Text(

'Bad Experience with customer service and adnfnasdfnasdfna ads f asdf as df asdf ',

style: TextStyle(

color: darktext.withAlpha(100),

fontSize: 13,

),

),

],

),

),

Icon(

Icons.chevron\_right\_rounded,

color: darktext,

size: 35,

),

],

),

),

),

Divider(

color: darktext.withAlpha(100),

),

],

);

}

}

class ReviewStar extends StatelessWidget {

final int a;

const ReviewStar({Key? key, this.a = 3}) : super(key: key);

@override

Widget build(BuildContext context) {

return Row(

children: [

for (int i = 0; i < a; i++)

Icon(

Icons.star,

size: 20,

color: darktext,

),

for (int i = 0; i < 5 - a; i++)

Icon(

Icons.star,

size: 20,

color: darktext.withAlpha(100),

),

],

);

}

}

**schedulescreen.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/servicetile.dart';

class scheduleScreen extends StatefulWidget {

const scheduleScreen({Key? key}) : super(key: key);

@override

State<scheduleScreen> createState() => \_scheduleScreenState();

}

class \_scheduleScreenState extends State<scheduleScreen> {

int totalcost = 0;

Map<String, List<DocumentSnapshot>> carservices = {};

final UserServices userServices = UserServices();

Future<Map<String, List<DocumentSnapshot>>> getdata() async {

QuerySnapshot querySnapshot =

await userServices.getunpaidservicewithuserkey();

totalcost = 0;

carservices.clear();

for (DocumentSnapshot documentSnapshot in querySnapshot.docs) {

carservices.putIfAbsent(documentSnapshot.get('carkey'), () => []);

carservices[documentSnapshot.get('carkey').toString()]

?.add(documentSnapshot);

totalcost += int.parse(documentSnapshot.get('serviceprice').toString());

}

return carservices;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

automaticallyImplyLeading: false,

title: Text(

"Schedules",

style: TextStyle(

fontWeight: FontWeight.bold,

color: darktext,

),

),

),

body: FutureBuilder(

future: getdata(),

builder: (context, snapshot) {

if (snapshot.hasData && snapshot.data!.length != 0) {

return Column(

children: [

Expanded(

child: ListView.builder(

padding: EdgeInsets.symmetric(vertical: 10),

physics: BouncingScrollPhysics(),

itemCount: snapshot.data!.length,

itemBuilder: (context, i) {

return section(

carkey: snapshot.data!.keys.elementAt(i),

documentSnapshot:

snapshot.data!.values.elementAt(i));

}),

),

Container(

padding: EdgeInsets.symmetric(vertical: 10),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Total Cost: ',

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

),

),

Text(

'₹ ' + totalcost.toString(),

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

],

),

),

],

);

} else if (snapshot.hasData && snapshot.data!.length == 0) {

return Center(

child: Text(

'No Services Booked Yet!!',

style: TextStyle(

fontSize: 20,

fontWeight: FontWeight.bold,

color: darktext,

),

),

);

} else {

return Center(

child: CircularProgressIndicator(),

);

}

}),

);

}

Widget section(

{required String carkey,

required List<DocumentSnapshot> documentSnapshot}) {

return Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

SizedBox(

height: 5,

),

Container(

padding: EdgeInsets.symmetric(horizontal: 15, vertical: 10),

child: FutureBuilder(

future:

userbase.doc(userkey).collection('cars').doc(carkey).get(),

builder: (context, snapshot) {

if (snapshot.hasData) {

return Text(

snapshot.data!.get('carmaker') +

' ' +

snapshot.data!.get('carmodel') +

": ",

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

color: darktext,

),

);

} else

return Container(

padding: EdgeInsets.symmetric(vertical: 10),

width: 60,

child: LinearProgressIndicator(),

);

}),

),

Column(

children: [

for (DocumentSnapshot ds in documentSnapshot)

serviceTile(documentSnapshot: ds),

],

),

],

);

}

}

**scheduleshop.dart**

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/data.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/models/shop\_model.dart';

import 'package:vehicle\_maintenance\_app/screens/mainscreens/homeparent.dart';

import 'package:vehicle\_maintenance\_app/screens/schedules\_screen/schedulereview.dart';

class scheduleShop extends StatefulWidget {

final ServiceModel serviceModel;

const scheduleShop({Key? key, required this.serviceModel}) : super(key: key);

@override

State<scheduleShop> createState() => \_scheduleShopState();

}

class \_scheduleShopState extends State<scheduleShop> {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: IconButton(

onPressed: () {

Navigator.pop(context);

},

icon: Icon(

CupertinoIcons.left\_chevron,

color: darktext,

),

),

title: Text(

"Schedule",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

margin: EdgeInsets.only(top: 15),

child: Column(

children: [

Row(

// mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

SizedBox(

width: 25,

),

Text(

'RECENT SHOPS',

style: subtitle,

),

Spacer(),

Icon(

Icons.more\_horiz\_rounded,

color: darktext,

),

SizedBox(

width: 30,

),

],

),

SizedBox(

height: 10,

),

Expanded(

child: ListView.builder(

physics: BouncingScrollPhysics(),

padding: EdgeInsets.only(bottom: 20),

itemCount: shopdata.length,

itemBuilder: (context, i) {

return shopTile(

shopmodel: shopdata[i],

);

},

),

),

],

),

),

);

}

Widget shopTile({required ShopModel shopmodel}) {

return Container(

margin: EdgeInsets.symmetric(vertical: 5, horizontal: 15),

child: ElevatedButton(

onPressed: () {

widget.serviceModel.shopmodel = shopmodel;

Navigator.pushNamed(context, '/schedulereview',

arguments: widget.serviceModel);

},

style: ElevatedButton.styleFrom(

backgroundColor: Colors.white,

// foregroundColor: maintheme,

elevation: 2,

surfaceTintColor: Colors.white,

padding: EdgeInsets.all(0),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

),

child: Container(

padding: EdgeInsets.symmetric(

vertical: 15,

horizontal: 15,

),

child: Row(

children: [

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

// 'Jiffy Lube',

shopmodel.shopname,

style: subtitle.copyWith(

fontSize: 15,

),

),

SizedBox(

height: 5,

),

ReviewStar(

a: 3,

),

SizedBox(

height: 5,

),

Text(

// '756, Barrington Road, Hanover Park 5245',

shopmodel.shopaddress,

style: subtitle.copyWith(

fontSize: 12,

color: darktext.withAlpha(100),

),

),

],

),

),

Icon(

Icons.chevron\_right\_rounded,

color: darktext,

size: 25,

),

],

),

),

),

);

}

}

**schedulesuccess.dart**

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

class scheduleSuccess extends StatelessWidget {

const scheduleSuccess({Key? key}) : super(key: key);

final String t2 =

"You'll receive a confirmation email from Jiffy Lube to confirm your requredted date and time.";

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

automaticallyImplyLeading: false,

scrolledUnderElevation: 0,

title: Text(

"Confirmation",

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: Container(

padding: EdgeInsets.symmetric(vertical: 25, horizontal: 30),

child: Column(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: [

Text(

'Thanks for Scheduling your next Service',

textAlign: TextAlign.center,

style: TextStyle(

fontSize: 24,

color: darktext,

fontWeight: FontWeight.bold,

),

),

Text(

t2,

textAlign: TextAlign.center,

style: TextStyle(

color: darktext,

fontSize: 18,

),

),

OutlinedButton(

onPressed: () {

Navigator.pop(context);

},

style: OutlinedButton.styleFrom(

foregroundColor: maintheme,

side: BorderSide(

color: maintheme,

),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

),

child: Container(

padding: EdgeInsets.symmetric(vertical: 15, horizontal: 15),

child: Text(

'Back to Dashboard',

style: TextStyle(

fontWeight: FontWeight.bold,

),

),

),

),

SizedBox(

height: 10,

),

Container(

height: 150,

width: 150,

child: Image.asset(

'assets/images/logo\_color.png',

),

),

SizedBox(

height: 20,

),

],

),

),

);

}

}

**servicemodel.dart**

import 'package:vehicle\_maintenance\_app/models/shop\_model.dart';

class ServiceModel {

String? carkey;

ShopModel? shopmodel;

String? servicename;

String? serviceprice;

String? day;

String? month;

String? year;

String? hours;

String? minutes;

String? notes;

String? paymentstatus;

ServiceModel({

this.carkey,

this.shopmodel,

this.servicename,

this.serviceprice,

this.day,

this.month,

this.year,

this.hours,

this.minutes,

this.notes = '',

this.paymentstatus = 'pending',

});

void printer() {

print(this.carkey);

print(this.shopmodel);

print(this.servicename);

print(this.serviceprice);

print(this.day);

print(this.month);

print(this.year);

print(this.hours);

print(this.minutes);

print(this.notes);

print(this.paymentstatus);

}

}

**servicetile.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

class serviceTile extends StatelessWidget {

final DocumentSnapshot documentSnapshot;

const serviceTile({Key? key, required this.documentSnapshot})

: super(key: key);

@override

Widget build(BuildContext context) {

return Container(

padding: EdgeInsets.symmetric(horizontal: 15, vertical: 5),

child: ElevatedButton(

onPressed: () {},

style: ElevatedButton.styleFrom(

foregroundColor: maintheme,

elevation: 7,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(16),

),

),

child: Container(

padding: EdgeInsets.symmetric(vertical: 15, horizontal: 10),

child: Row(

children: [

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

documentSnapshot.get('servicename').toString(),

maxLines: 1,

overflow: TextOverflow.ellipsis,

style: subtitle,

),

Text(

'at ' + documentSnapshot.get('shopname').toString(),

maxLines: 1,

overflow: TextOverflow.ellipsis,

style: subtitle.copyWith(

fontSize: 13,

fontWeight: FontWeight.normal,

),

),

SizedBox(

height: 5,

),

Row(

children: [

Text(

documentSnapshot.get('servicedate').toString(),

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

SizedBox(

width: 15,

),

Text(

documentSnapshot.get('servicetime').toString(),

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

],

),

],

),

),

SizedBox(

width: 20,

),

Column(

children: [

Text(

'₹ ' + documentSnapshot.get('serviceprice').toString(),

style: TextStyle(

fontWeight: FontWeight.bold,

color: darktext,

fontSize: 18,

),

),

SizedBox(

height: 10,

),

paidBadge(

paid: (documentSnapshot.get('paymentstatus').toString() ==

'pending')

? (false)

: (true)),

],

),

],

),

),

),

);

}

}

paidBadge({bool paid = true}) {

if (paid == true) {

return Row(

children: [

Icon(

Icons.check\_circle\_outline\_rounded,

color: Colors.green,

size: 15,

),

SizedBox(

width: 5,

),

Text(

'Paid',

style: TextStyle(

fontSize: 13,

color: Colors.green,

fontWeight: FontWeight.bold,

),

),

],

);

} else {

{

return Row(

children: [

Icon(

Icons.not\_interested\_rounded,

color: Colors.red,

size: 15,

),

SizedBox(

width: 5,

),

Text(

'Not Paid',

style: TextStyle(

fontSize: 13,

color: Colors.red,

fontWeight: FontWeight.bold,

),

),

],

);

}

}

}

**shop\_model.dart**

class ShopModel {

final String shopname;

final String shopaddress;

final String shopphone;

ShopModel({

required this.shopname,

required this.shopaddress,

required this.shopphone,

});

}

**signup.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/screens/loginpage.dart';

import 'package:vehicle\_maintenance\_app/services/user\_auth.dart';

class SignUp extends StatefulWidget {

const SignUp({Key? key}) : super(key: key);

@override

State<SignUp> createState() => \_SignUpState();

}

class \_SignUpState extends State<SignUp> {

TextEditingController name = TextEditingController();

TextEditingController email = TextEditingController();

TextEditingController password = TextEditingController();

String errortext = '';

bool obscurepassword = true;

Authentication \_authentication = Authentication();

bool loading = false;

bool validate() {

if (email.text != '' && name.text != '' && password.text != '') {

//signup

return true;

} else {

setState(() {

errortext = '';

if (email.text == '') {

errortext += 'Enter Email Id\n';

}

if (name.text == '') {

errortext += 'Enter Name\n';

}

if (password.text == '') {

errortext += 'Enter Password Id\n';

}

});

return false;

}

}

showSuccess() async {

await showDialog(

context: context,

barrierDismissible: false,

builder: (context) => AlertDialog(

title: Text('Sign Up Successfull'),

content: Text('You can Login Now'),

contentPadding: EdgeInsets.all(30),

actions: [

TextButton(

onPressed: () {

Navigator.pop(context);

},

child: Text('Go to Login Page'),

),

],

),

);

}

signup() async {

setState(() {

loading = true;

});

var result =

await \_authentication.createUser(name.text, email.text, password.text);

if (result.runtimeType == bool && result == true) {

//success

await showSuccess();

Navigator.pushReplacementNamed(context, '/login');

} else if (result.toString().contains('email-already-in-use')) {

setState(() {

errortext = 'Email Already in Use\nTry Logging in';

});

} else {

setState(() {

errortext = 'Sign up successfully Failed';

});

}

setState(() {

loading = false;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: SafeArea(

child: Container(

padding: EdgeInsets.symmetric(horizontal: 15),

child: Column(children: [

Row(

mainAxisAlignment: MainAxisAlignment.end,

children: [

TextButton(

onPressed: () async {

Navigator.pushReplacementNamed(context, '/login');

},

child: Text(

"Login",

style: TextStyle(

fontWeight: FontWeight.bold,

),

),

),

],

),

Expanded(

child: SingleChildScrollView(

physics: BouncingScrollPhysics(),

child: Container(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.center,

children: [

SizedBox(

height: 50,

),

Row(),

// Container(

// child: Icon(

// Icons.car\_crash\_sharp,

// color: Colors.redAccent,

// size: 100,

// ),

// ),

Container(

height: 150,

width: 150,

child: Image.asset(

'assets/images/logo\_color.png',

),

),

SizedBox(

height: 10,

),

Text(

"VehiCare",

style: TextStyle(

color: maintheme,

fontWeight: FontWeight.bold,

fontSize: 35,

),

),

SizedBox(

height: 30,

),

Container(

height: 60,

child: TextField(

controller: name,

style: TextStyle(

fontSize: 18,

),

decoration: InputDecoration(

hintText: 'Name',

hintStyle: TextStyle(

color: Colors.grey,

fontSize: 18,

),

focusedBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 1,

),

),

enabledBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 0.5,

),

),

),

),

),

SizedBox(

height: 20,

),

Container(

height: 60,

child: TextField(

controller: email,

style: TextStyle(

fontSize: 18,

),

decoration: InputDecoration(

hintText: 'Email',

hintStyle: TextStyle(

color: Colors.grey,

fontSize: 18,

),

focusedBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 1,

),

),

enabledBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 0.5,

),

),

),

),

),

SizedBox(

height: 20,

),

Row(

children: [

Expanded(

child: Container(

height: 60,

child: TextField(

controller: password,

style: TextStyle(

fontSize: 18,

),

obscureText: obscurepassword,

obscuringCharacter: '\*',

decoration: InputDecoration(

hintText: 'Password',

hintStyle: TextStyle(

color: Colors.grey,

fontSize: 18,

),

focusedBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 1,

),

),

enabledBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(12),

borderSide: BorderSide(

color: maintheme,

width: 0.5,

),

),

),

),

),

),

SizedBox(

width: 5,

),

IconButton(

onPressed: () {

setState(() {

obscurepassword = !obscurepassword;

});

},

icon: (obscurepassword)

? (Icon(Icons.visibility\_outlined))

: (Icon(Icons.visibility\_off\_outlined)),

),

],

),

SizedBox(

height: 5,

),

SizedBox(

height: 30,

),

TextButton(

onPressed: (loading == true)

? (null)

: (() {

if (validate() == true) {

signup();

}

}),

style: TextButton.styleFrom(

backgroundColor: maintheme,

foregroundColor: Colors.white,

padding: EdgeInsets.zero,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

),

child: Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.center,

children: [

(loading == true)

? Container(

padding: EdgeInsets.symmetric(vertical: 10),

child: (CircularProgressIndicator(

color: Colors.white,

)),

)

: (Container(

height: 60,

alignment: Alignment.center,

child: Text(

"Sign Up",

style: TextStyle(

color: Colors.white,

fontSize: 20,

),

),

))

],

),

),

SizedBox(

height: 50,

),

Text(

errortext,

style: TextStyle(

color: Colors.red,

fontWeight: FontWeight.bold,

),

),

],

),

),

),

),

]),

),

),

);

}

}

**upcoming\_appointments.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicle\_maintenance\_app/global.dart';

import 'package:vehicle\_maintenance\_app/services/user\_services.dart';

import 'package:vehicle\_maintenance\_app/widgets/servicetile.dart';

class upcomingAppointmentScreen extends StatefulWidget {

final String carkey;

final String title;

upcomingAppointmentScreen(

{Key? key,

this.carkey = 'IgoC6hm8VmVNKjlleTET',

this.title = 'Upcoming Appointments'})

: super(key: key);

@override

State<upcomingAppointmentScreen> createState() =>

\_upcomingAppointmentScreenState();

}

class \_upcomingAppointmentScreenState extends State<upcomingAppointmentScreen> {

int totalcost = 0;

final UserServices userServices = UserServices();

Future<List<DocumentSnapshot>> getdata() async {

QuerySnapshot querySnapshot =

await userServices.getunpaidservicewithcarkey(carkey: widget.carkey);

List<DocumentSnapshot> documentsnapshots = [];

totalcost = 0;

for (DocumentSnapshot documentSnapshot in querySnapshot.docs) {

documentsnapshots.add(documentSnapshot);

totalcost += int.parse(documentSnapshot.get('serviceprice').toString());

}

return documentsnapshots;

}

@override

void initState() {

// TODO: implement initState

super.initState();

getdata();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.white,

elevation: 0,

centerTitle: true,

scrolledUnderElevation: 0,

leading: IconButton(

onPressed: () {

Navigator.pop(context);

},

icon: Icon(

CupertinoIcons.left\_chevron,

color: darktext,

),

),

title: Text(

(widget.title == '') ? ("Upcoming Appointments") : (widget.title),

style: TextStyle(

color: darktext,

fontWeight: FontWeight.bold,

),

),

),

body: FutureBuilder(

future: getdata(),

builder: (context, snapshot) {

if (snapshot.hasData && snapshot.data!.length != 0) {

return Column(

children: [

Expanded(

child: ListView.builder(

physics: BouncingScrollPhysics(),

itemCount: snapshot.data!.length,

itemBuilder: (context, i) {

return serviceTile(

documentSnapshot: snapshot.datai],

);

}),

),

Container(

padding: EdgeInsets.symmetric(vertical: 10, horizontal: 15),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Total Cost: ',

style: TextStyle(

fontSize: 16,

fontWeight: FontWeight.bold,

),

),

Text(

'₹ ' + totalcost.toString(),

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

Spacer(),

OutlinedButton(

onPressed: () {

Navigator.pushNamed(

context,

'/paymentscreen',

arguments: snapshot.data!,

);

},

style: OutlinedButton.styleFrom(

backgroundColor: Colors.white,

foregroundColor: Colors.red,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(12),

),

side: BorderSide(

color: Colors.red,

width: 1,

),

),

child: Text(

'Pay All',

style: TextStyle(

color: Colors.red,

fontSize: 15,

),

),

),

],

),

),

],

);

} else if (snapshot.hasData && snapshot.data!.length == 0) {

return Center(

child: Text(

'No Services Booked Yet!!',

style: TextStyle(

fontSize: 20,

fontWeight: FontWeight.bold,

color: darktext,

),

),

);

} else {

return Center(

child: CircularProgressIndicator(),

);

}

}),

);

}

}

**user\_auth.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

class Authentication {

createUser(String name, String email, String password) async {

UserCredential user;

try {

user = await firebaseAuth.createUserWithEmailAndPassword(

email: email, password: password);

} catch (e) {

print(e);

return e;

}

if (user.additionalUserInfo!.isNewUser) {

userbase.doc(user.user!.uid.toString()).set({

'name': name,

'cars': {},

'services': {},

});

}

return true;

}

loginUser(String email, String password) async {

UserCredential user;

try {

user = await firebaseAuth.signInWithEmailAndPassword(

email: email, password: password);

//getting instance for local storage

SharedPreferences sharedPreferences =

await SharedPreferences.getInstance();

//getting snapshot for username

DocumentSnapshot documentSnapshot =

await userbase.doc(user.user!.uid).get();

// storing userkey and user name in local variable

username = documentSnapshot.get('name');

userkey = user.user!.uid.toString();

// storing userkey and user name in local storage

sharedPreferences.setString('userkey', user.user!.uid);

sharedPreferences.setString('username', documentSnapshot.get('name'));

} catch (e) {

print(e);

return e;

}

// print(user.user!.uid.toString());

return user;

// print(user);

}

}

**user\_services.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:vehicle\_maintenance\_app/models/servicemodel.dart';

import 'package:vehicle\_maintenance\_app/services/constants.dart';

class UserServices {

addnewcar({required String carmaker, required String carmodel}) async {

await userbase.doc(userkey).collection('cars').add({

'carmaker': carmaker,

'carmodel': carmodel,

});

print('Success');

}

addSchedule({required ServiceModel serviceModel}) async {

var result = await userbase.doc(userkey).collection('services').add({

'carkey': serviceModel.carkey,

'shopname': serviceModel.shopmodel!.shopname,

'servicename': serviceModel.servicename,

'serviceprice': serviceModel.serviceprice.toString(),

'servicedate': serviceModel.day.toString() +

'/' +

serviceModel.month.toString() +

'/' +

serviceModel.year.toString(),

'servicetime':

serviceModel.hours.toString() + ':' + serviceModel.minutes.toString(),

'servicenotes': serviceModel.notes,

'paymentstatus': serviceModel.paymentstatus,

});

return result;

}

Future<QuerySnapshot> getcars() async {

QuerySnapshot querySnapshot =

await userbase.doc(userkey).collection('cars').get();

return querySnapshot;

}

Future<QuerySnapshot> getserviceswithcarkey({required String carkey}) async {

QuerySnapshot snapshot = await userbase

.doc(userkey)

.collection('services')

.where('carkey', isEqualTo: carkey)

.get();

return snapshot;

}

Future<QuerySnapshot> getserviceswithuserkey() async {

QuerySnapshot querySnapshot =

await userbase.doc(userkey).collection('services').get();

return querySnapshot;

}

Future<QuerySnapshot> getunpaidservicewithcarkey(

{required String carkey}) async {

QuerySnapshot querySnapshot = await userbase

.doc(userkey)

.collection('services')

.where('carkey', isEqualTo: carkey)

.where('paymentstatus', isEqualTo: 'pending')

.get();

return querySnapshot;

}

Future<QuerySnapshot> getunpaidservicewithuserkey() async {

QuerySnapshot querySnapshot = await userbase

.doc(userkey)

.collection('services')

.where('paymentstatus', isEqualTo: 'pending')

.get();

return querySnapshot;

}

deleteCar({required String carkey}) async {

// to delete document from car collection

await userbase.doc(userkey).collection('cars').doc(carkey).delete();

print('user deleted');

// to delete its services

QuerySnapshot querySnapshot = await userbase

.doc(userkey)

.collection('services')

.where('carkey', isEqualTo: carkey)

.get();

for (DocumentSnapshot doc in querySnapshot.docs) {

await userbase.doc(userkey).collection('services').doc(doc.id).delete();

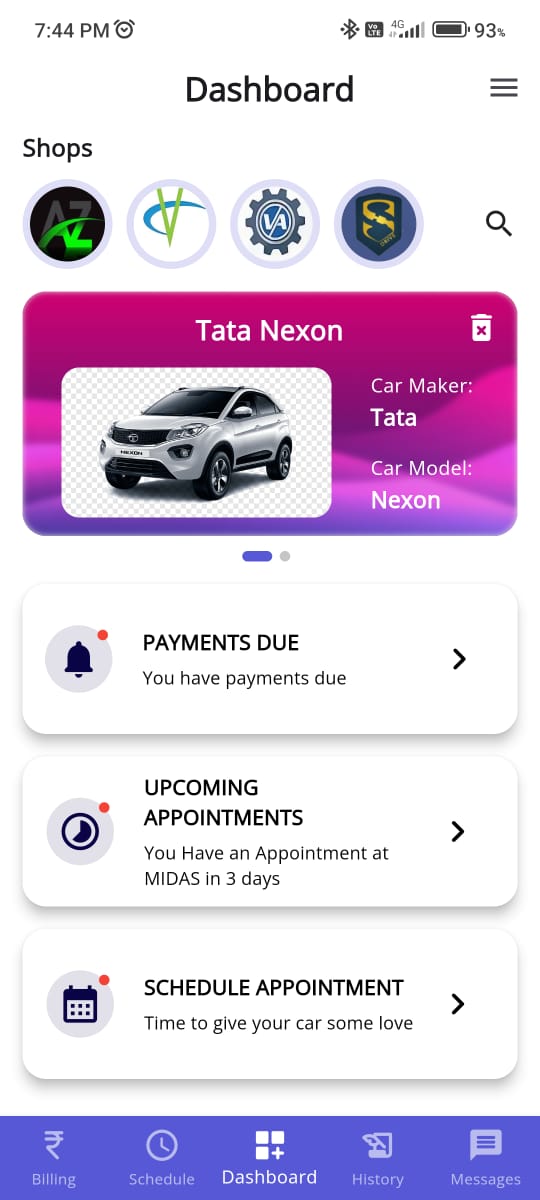
}

print('services deleted');

}

}

**IMPLEMENTATION OF APP**

 ![Graphical user interface

Description automatically generatedA screenshot of a phone

Description automatically generated with medium confidence

Fig 12.8 – home page fig 12.9 – add vehicle fig 12.10 – delete vehicle

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Fig 12.11 – payment fig 12.12 – payment 2 fig 12.13 – schedules

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

fig 12.14 – schedule 2 fig 12.15 – schedule 3 fig 12.16 - history

**CONCLUSION**

In conclusion, the Vehicle Maintenance System project has been successfully completed, and the resulting system has met all the functional and non-functional requirements identified at the outset of the project. The system provides an efficient and effective way for fleet managers to manage their vehicles, schedule maintenance and repairs, and track maintenance costs and vehicle usage.

The RAD model was used for the project, which allowed for rapid development and iteration, and facilitated effective communication and collaboration between the development team and the stakeholders. The use of agile methodologies, including regular sprint reviews and retrospectives, helped to ensure that the project remained on track and that any issues were identified and addressed in a timely manner.

Throughout the project, a user-centered design approach was adopted, with a focus on creating a system that is intuitive and easy to use for drivers, maintenance personnel, and fleet managers. The system's interface is designed to be user-friendly, with clear navigation and easily accessible information, and the system is compatible with a wide range of devices and software platforms.

In conclusion, the Vehicle Maintenance System project has been a success, and the resulting system is a valuable tool for managing fleet maintenance and repairs.

**REFERENCES**

1. "Vehicle Maintenance Software," Fleetio, accessed April 22, 2023, https://www.fleetio.com/features/maintenance.
2. "A User-Centered Design Approach to Vehicle Maintenance Systems," International Journal of Industrial Ergonomics 73 (2019): 44-54.
3. "Agile Project Management," Agile Alliance, accessed April 22, 2023, https://www.agilealliance.org/agile101/agile-project-management/.
4. "Rapid Application Development," Gartner IT Glossary, accessed April 22, 2023, https://www.gartner.com/en/information-technology-glossary/rapid-application-development-rad.
5. "Designing User Interfaces for Fleet Management Systems," Proceedings of the 11th International Conference on Human-Computer Interaction (HCI International 2019), Orlando, FL, July 26-31, 2019.
6. "Scalable and Secure Vehicle Maintenance Systems," IEEE Transactions on Intelligent Transportation Systems 20, no. 3 (2019): 1010-1023.
7. "Software Engineering Principles and Practices," Roger S. Pressman and Bruce R. Maxim, John Wiley & Sons, 2015.