Project requirements - Gearshare

Introduction

This document provides audiences with a comprehensive overview of the project aimed at enhancing the data management system for Gear Share.

|  |  |
| --- | --- |
| Project Background | Gear Share, a local equipment rental company that also provides repair services, faces several operational challenges due to its rapid growth. Currently, accessing information between Booqable and Airtable requires users to switch between platforms, leading to inefficiencies in workflow. Additionally, Booqable’s email capabilities are limited to its predefined templates, restricting the ability to send customized and automated communications. Although Zapier automates some processes, many tasks still require manual intervention. As a result, Gear Share is seeking to develop an integrated software solution to streamline workflows, enhance automation, and improve overall service efficiency. |
| Scope | This project focuses solely on enhancing the software layer and will not address any issues related to hardware. |
| Objective | This project will develop a comprehensive solution to enable the company to manage all its services with additional functionalities. |
| System Environments | This project will deliver a prototype that operates seamlessly on both web and iOS platforms. |
| Stakeholders | The client’s team consists entirely of nontechnical professionals, and the development team is composed of members with formal education in computer science. |
| Product Perspective | The product of this project will be a prototyping software that integrates the current services provided by Airtable and Booqable. It will enable users to manage inventory and track repair records more efficiently with additional functions. |
| Constraints | This project will deliver a prototype only and will not include any future maintenance or updates. Additionally, the prototype may not fully replace the existing system initially and may require further modifications to achieve complete functionality. |

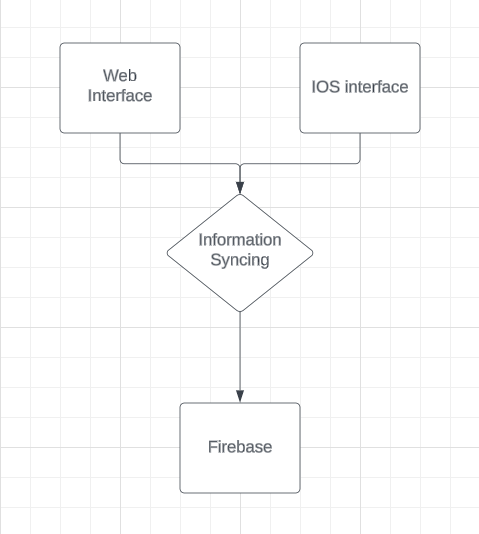
Functional Requirements

|  |  |
| --- | --- |
| Data management and integration | The management system will collect all data from Airtable and Booqable and ensure data synchronization between both platforms. Users will be able to manage inventory, rental records, user information, and repair tickets through the system. |
| Web app & IOS app | The web app and iOS app will provide a consistent experience and functionality, with full data synchronization between platforms. |
| Automatic process | The management system will automate various customer service processes, including data migration, information synchronization, quoting the services and report generation. Emails sent to users will be automatically generated using a customized template that can be edited by user. A chatbot feature will also be added to support customer service interactions. In addition, the payment of customer will also be managed automatically by the system. |
| User management | The management system will support various user roles with different levels of authorization. Users will be able to view and edit their personal information based on their assigned permissions. |

Nonfunctional Requirements

|  |  |
| --- | --- |
| User friendly interface | Given that the client team consists entirely of nontechnical professionals, the product will feature a graphical interface that only requires basic skills such as clicking and typing. |
| Security | Data storage and transmission will be secured through encryption mechanisms. Information access will be controlled based on user roles, with logs generated to monitor user activity. |

Technology Stack



|  |  |
| --- | --- |
| Frontend | The web interface and iOS app will provide users with access to all functionalities of the management system. |
| Middleware | The information syncing process will collect and integrate data from Airtable and Booqable through their APIs. |
| Backend | The initial development is designed to use Firebase to store user, inventory, and repair data. |

Data model

A screenshot of a computer

Description automatically generated

The data model for this project is structured around five primary components: Users, Activity Logs, Repair Tickets, and Rental Records. Data will be interconnected and shared across these components through the unique identifiers.

Acceptance Criteria

|  |  |
| --- | --- |
| User management | Users can register and log in using a username and password. Once user create an account, the user profile will be created. Administrators can manage user accounts by creating, deleting, or editing entries and monitoring user activity. The system will enforce access control based solely on user roles. |
| Data integration | The system can collect data from both Airtable and Booqable, ensuring seamless information synchronization across platforms. |
| Equipment management | Users can view the status and pricing of equipment, as well as add or remove equipment entries. Any attribute changes will be reflected in both Airtable and Booqable. |
| Rental Management | Users can view and edit equipment records, with certain services being more automated. Any changes to attributes will be automatically synchronized with Airtable and Booqable. |
| Repair Ticket Management | Customers can submit repair tickets, while maintenance staff can view assigned tasks and update their work progress. The system will automatically notify users of status changes and send out quota through autogenerated emails. |

API

The following are APIs the development team may use in production.

|  |  |
| --- | --- |
| Airtable API | This API can be used to integrate data in Airtable with any external system.  Source: [Introduction - Airtable Web API](https://airtable.com/developers/web/api/introduction) |
| Bookqable API | This API can be used to view and editing the information from Bookquable.  Source: [Introduction – Booqable API](https://developers.booqable.com/#introduction) |
| Gmail API | This API can be used to automatic the process of sending email.  Source: [Sending Email | Gmail  |  Google for Developers](https://developers.google.com/gmail/api/guides/sending) |
| Venmo API | This API can be used to automatic the payment process through Venmo.  Source: [venmo.com/docs/overview/](https://venmo.com/docs/overview/) |
| Square API | This API can be used to automatic the payment process through Square.  Source: [Square APIs & SDKs: Build Custom Solutions on Square (squareup.com)](https://developer.squareup.com/us/en?device=c&gad_source=1&gclid=CjwKCAjw9eO3BhBNEiwAoc0-jT8S7VsuMZIW_rjDfmf3BRvLqdZOEd9ULaPWhT0e8vX58xhXqFBGThoCe-YQAvD_BwE&gclsrc=aw.ds&kw=square+api&kwid=p62988685545&matchtype=e&pcrid=636642237007&pdv=c&pkw=square+api&pmt=e&pub=GOOGLE) |

Stakeholders

|  |  |
| --- | --- |
| Josh Bossin (Gearshare) | josh@mainegearshare.org |
| Emily Mackeown (Gearshare) | emily@mainegearshare.org |
| Kevin Chen (development team) | chen.kevin4@northeastern.edu |
| Comus Hardman (development team) | hardman.c@northeastern.edu |
| Vageesh Kudutini Ramesh  (development team) | [kudutiniramesh.v@northeastern.edu](mailto:kudutiniramesh.v@northeastern.edu) |
| Yixiao Wu (development team) | wu.yixia@northeastern.edu |