AN ALL-IN-ONE EQUESTRIAN MANAGEMENT APP DESIGNED FOR TRAINERS, RIDERS AND GROOMS THAT HELPS WITH BARN ORGANIZATION

Jiajin Li [1], Rodrigo Onate [2]

[1] Marlborough School, 250 South Rossmore Avenue, Los Angeles, CA 90004
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

Barn management often struggles with poor communication, outdated scheduling, and scattered information, leading to confusion and missed lessons. My app, HorsiTask, solves this problem by offering a clear, all-in-one platform for riders, trainers, and grooms to manage schedules, tasks, and horse care in real time [6]. The app uses Firebase for data syncing and Flutter for the interface. Two key challenges were preventing double bookings and ensuring clean data input [7]. We addressed these through experiments: one tested the scheduling system under stress, and another tested form validation by submitting flawed entries. Both showed that HorsiTask can block invalid inputs and maintain accurate booking records. The most important result was that our system was held up under pressure and caught most errors. HorsiTask improves on past barn solutions by reducing miscommunication and confusion through automation and clear design. It's a tool that makes barn life easier and more efficient for everyone involved.

KEYWORDS

Horses, Equestrian, Management Application, All-in-One

1. INTRODUCTION

Despite advances in many industries, barn management within the equestrian world remains largely inefficient and disorganized. Many barns still depend on outdated systems such as whiteboards, handwritten schedules, and scattered text messages. These methods are difficult to update, easily forgotten, and often lead to significant miscommunication among riders, trainers, and staff. For example, it is not uncommon for riders to arrive at the barn only to discover their horse is not ready, or to miss out on shows and training sessions because crucial information was not communicated or displayed clearly. This is not an isolated problem. According to a survey conducted by The Chronicle of the Horse, 78 percent of riders reported frustration due to miscommunications at their barn, while 64 percent admitted to missing at least one event as a result of scheduling confusion [1]. The challenges only increase in larger facilities where more horses, riders, and grooms are involved, making it even harder to keep track of tasks and schedules. Trainers often have to repeat information multiple times or spend additional effort coordinating with their clients regarding their availability for lessons or competitions. The growing demand for efficiency and transparency has driven research in digital barn management. According to Research Business Insights, digital management tools have proven to enhance operational efficiency and rider satisfaction. The global horse software market is reflecting this trend, with a valuation of $0.22 billion in 2024 and a projected increase to $0.47 billion by 2033, indicating widespread adoption of digital

David C. Wyld et al. (Eds): CMLA, CSEIT, NLPD, GRAPH-HOC, SPM, NeTCoM, NCS, ArIT, DMSE – 2025
pp.153 -162, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151313

solutions in equestrian facilities [2]. Ultimately, the lack of an effective system negatively affects riders, trainers, and barn staff, highlighting the urgent need for a comprehensive and user-friendly management solution like Horsitask.

The first methodology focused on organized barn management through simple routines and tools. It worked well for scheduling and daily planning but didn't help much with training staff or managing behavior. Horsi Task improves this by offering real-time updates and communication between users.

The second methodology highlighted the importance of day-to-day bio security practices. It was useful because it gave caretakers clear steps to follow, but it depended too much on people remembering everything and didn't cover tech integration. Horsi Task builds on this by automating updates and helping users avoid mistakes.

The third method emphasized the value of barn culture and communication. It helped build better teamwork and smoother routines, but it needed strong leadership and didn't solve technical problems. Horsi Task strengthens this by offering a shared, structured platform where everyone stays informed, making teamwork and communication easier through one tool.

Horsitask will solve the problem at hand by creating a simple, easy-to-use system that keeps trainers, riders, and grooms (barn faculty) informed about their specific tasks, ensuring that barns run smoothly.

Horsitask focuses on all barn events, horse care schedules, and lesson plans. With Horsitask, riders can easily view their planned schedule, request a lesson with a trainer, and check for current events both at and outside the barn. Trainers can manage their schedules by accepting or declining lesson requests, post events on the bulletin board to keep users informed and oversee classes—all from the convenience of their smart phone. Grooms will have clear, accessible lists of daily responsibilities, reduce confusion and ensure that every horse receives the proper care before each lesson.

Barn management requires organization, record-keeping, cleanliness, and effective communication. Horsitask meets these requirements by streamlining key components into a single, user-friendly platform [14]. Unlike traditional paper schedules, fragmented communication through calls, text messages, or white boards, Horsitask integrates scheduling, messaging, and task management into one seamless system. This provides trainers, riders, and grooms with real-time updates, minimizing miscommunication, scheduling conflicts, and overlooked tasks.

This solution is more effective than other methods because it simplifies barn operations, eliminates the inefficiencies of manual scheduling, and improves coordination among all barn staff and users. By offering a structured, intuitive interface, Horsitask not only makes barn management more efficient but also enhances the overall experience for everyone involved.

The goal of Experiment A was to test whether Horsi Task's lesson scheduling system could prevent double booking. By having multiple users try to book the same time slot with the same trainer using test accounts, the experiment confirmed that only one booking was allowed, showing the system handled conflicts correctly. Experiment B focused on form validation. We created scenarios with missing or incorrect inputs and measured how many were successfully blocked. Most invalid entries were caught, with only a few slipping through. The design of both experiments used control groups to verify results. Key findings showed that Firebase's atomic transactions helped prevent scheduling conflicts and that Flutter's

validation logic mostly worked but could be improved. These experiments validated that the system is generally reliable under real-world conditions, with some room for improvement in edge cases. Both experiments supported the idea that HorsiTask can improve organization and prevent common user errors in barn management.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Establish Safe Access Control

One of the major problems in authentication is to establish safe access control and to distinguish between normal users and teachers. Unauthorized access would violate confidential data, and session management issues could lead to users being logged out instantly. For prevention against these, I could use Firebase Authentication along with role-based access control, where user roles would be stored in Firestore [8]. This would grant different privileges for different users. In addition, authentication state listeners can be used to help session persistence in order to preserve users logged on securely. Multi-factor authentication (MFA) can be used as an extra security factor, reducing the possibility of unauthorized access [9].

2.2. System

It's hard to efficiently manage event scheduling without facing double bookings and keeping devices up-to-date in real time. Only instructors can create events, and they will require a good system to handle bookings, accept or delete classes, and avoid scheduling conflicts. When two students want the same time slot, this may create a conflict that might confuse people. To avoid such conflicts, I could use Firestore transactions to check for availability before confirming lessons [15]. Firestore's snapshot listeners would ensure that all users see real-time updates on their calendars, allowing them to track their activities seamlessly. Additionally, validation rules could enforce correct data entry, ensuring that instructors cannot schedule overlapping classes. Notifications could also be implemented to inform students when their classes are accepted, canceled, or rescheduled.

2.3. Data Inaccuracies

Handling horse and lesson data includes maintaining the data integrity and avoiding unauthorized modifications. In case access is not properly controlled, regular users may alter horse records or interfere with lesson booking, leading to data inaccuracies. I can prevent this by using Firebase Security Rules so that horse records can be added, updated, or deleted by instructors alone. Since more than one student may book the same lesson time, concurrency may be an issue, leading to conflicts in scheduling. Firestore batch writes would help by ensuring lesson bookings are done sequentially, hence preventing double booking. Data validation checks on the data would also enforce proper formatting of horse information so that there is uniformity [10]. For easier organization, I would implement a formalized lesson request system where students request lessons and instructors approve them manually to prevent scheduling conflicts.

3. SOLUTION

The three major components that helped me link my program together was authentication system, Event and calendar management and horse and lesson management. When the

application launches the user arrives at the login page, the users are given the options to either login or sign up. If the user is new and has to sign up they would land on the profile page, requiring them to fill out their own information. After a firebase authentication, the user will land on the home page, showing the calendar with the present date and the events that are occurring [11]. Through the bottom navigation bar, the users get to switch around the pages. The pages include the horse page, the events page and their profile page. The horse page is where the user could create their own horse by clicking on the add button on the top right. When creating a horse, the user gets to upload a picture of their horse and fill in information such as the horse's name, gender, age etc. If the user made a mistake, they can click edit profile and make any adjustments to their horse. When done, by clicking on the three dots on the top right the users could either delete their horse, or schedule a lesson with a trainer. A small window will pop up and ask for the preferred time. After scheduling the lesson, it will appear on the calendar on the landing page. All of these features are made with flutter and firebase.

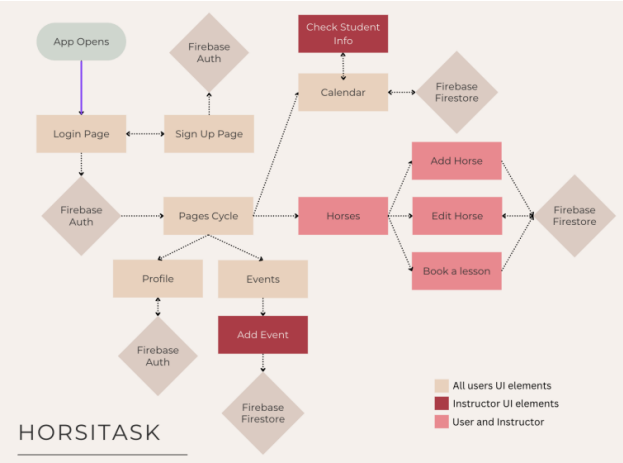


Figure 1. Overview of the solution

The authentication system was created using firebase authentication service. The purpose of having the authentication system is to keep track of each and every individual's information, making sure each user has their own account. This component relies on firebase to save the user's email, password, and to create a user id for accessing data from firebase firestore service.

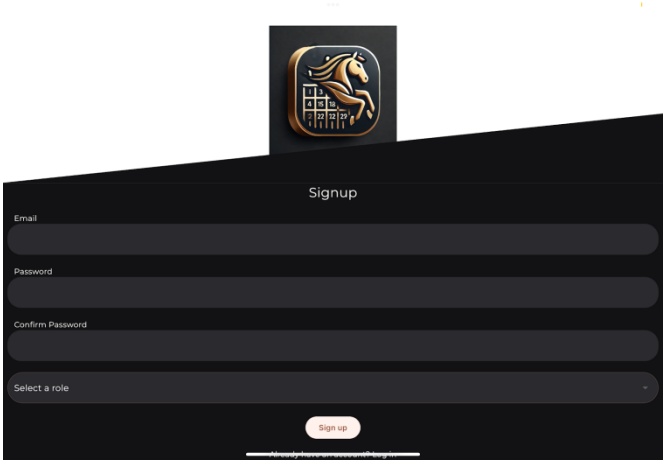


Figure 2. Screenshot of the sign up page

```dart
// Function to sign up user and save role
Future<void> _signupUser() async {
  if (_selectedRole == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Please select a role")),
    );
    return;
  }

  if (_passwordController.text != _confirmPasswordController.text) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Passwords do not match")),
    );
    return;
  }

  setState(() => _isLoading = true);

  try {
    UserCredential userCredential =
        await _auth.createUserWithEmailAndPassword(
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
    );

    // Save user role in Firestore
    await _firestore.collection("users").doc(userCredential.user!.uid).set({
      "email": _emailController.text.trim(),
      "role": _selectedRole,
      "uid": userCredential.user!.uid,
      'name': '',
    });

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("User created successfully")),
    );
    Navigator.pop(context);
```

Figure 3. Screenshot of code 1

This code runs when a user signs up in the app. First, it checks if the user picked a role. If not, it shows a message asking them to choose one. Then, it checks if the password and the confirmed password match. If everything is okay, it tries to create a new account using the email and password the user typed.

If the account is created successfully, it saves the user's email, role, and ID in Firestore (a database). It also shows a message that the account was made and takes the user back to the previous screen. The code uses input from the email, password, and role fields. It helps connect the app to the backend (Firebase), where the user's information is stored.

The event component of Horsitask is built using Firebase Firestore, allowing for real-time updates and seamless communication across the entire barn. Trainers and managers can create events like horse shows, clinics, vet visits, or farrier appointments, complete with dates, times, horse assignments, and notes. As soon as an event is added or changed, everyone,riders, grooms, and staff sees the update instantly. This eliminates confusion, missed messages, and the need for group chats or whiteboard reminders.

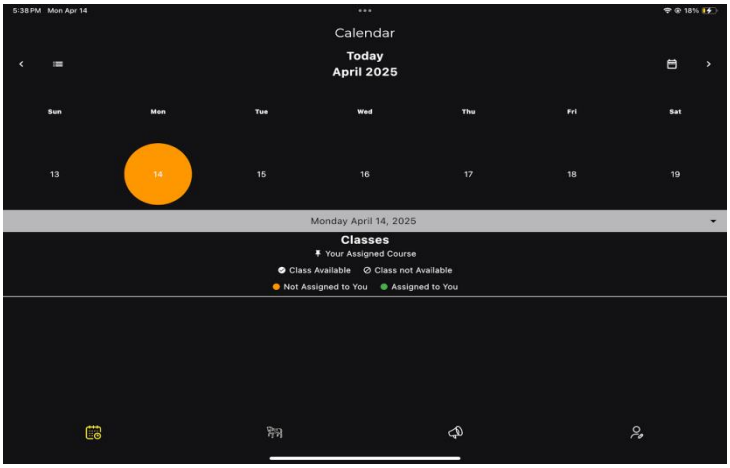
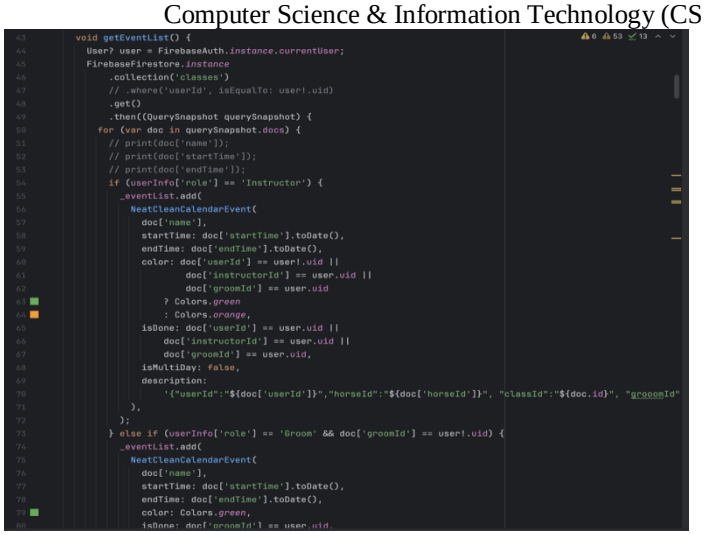


Figure 4. Screenshot of calendar



```
void getEventList() {
  User? user = FirebaseAuth.instance.currentUser;
  FirebaseFirestore.instance
      .collection('classes')
      // .where('userId', isEqualTo: user!.uid)
      .get()
      .then((QuerySnapshot querySnapshot) {
    for (var doc in querySnapshot.docs) {
      // print(doc['name']);
      // print(doc['startTime']);
      // print(doc['endTime']);
      if (userInfo['role'] == 'Instructor') {
        _eventList.add(
          NeatCleanCalendarEvent(
            doc['name'],
            startTime: doc['startTime'].toDate(),
            endTime: doc['endTime'].toDate(),
            color: doc['userId'] == user!.uid ||
                    doc['instructorId'] == user.uid ||
                    doc['groomId'] == user.uid
                ? Colors.green
                : Colors.orange,
            isDone: doc['userId'] == user.uid ||
                doc['instructorId'] == user.uid ||
                doc['groomId'] == user.uid,
            isMultiDay: false,
            description:
                '{"userId":"${doc['userId']}","horseId":"${doc['horseId']}", "classId":"${doc.id}", "groomId"
          ),
        );
      } else if (userInfo['role'] == 'Groom' && doc['groomId'] == user!.uid) {
        _eventList.add(
          NeatCleanCalendarEvent(
            doc['name'],
            startTime: doc['startTime'].toDate(),
            endTime: doc['endTime'].toDate(),
            color: Colors.green,
```

Figure 5. Screenshot of code 2

This Dart function, getEventList(), gets class events from Firestore 'classes' collection and filters the events based on the role and UID of the current user. It first retrieves the logged-in user through Firebase Authentication, then retrieves the 'classes' documents. For each document, if the user is an 'Instructor' and their UID matches any of the userId, instructorId, or groomId fields, a NeatCleanCalendarEvent is added to _eventList. The event has properties such as name, startTime, endTime, and isDone. The event color is green if the user is involved, otherwise orange. A description is also generated with corresponding IDs. If the user's role is 'Groom' and the UID is the same as the document's groomId, then an event of similar type is added. This technique helps populate a calendar view with only events a user is associated with, meaning every user views their own personalized information based on where they sit in the system. It also ensures events get colored and formatted appropriately for viewing.

The Horses component in Horsitask is designed to keep all horse-related information organized and up to date [12]. Both instructors and riders can access this section to add new horses, edit details like training notes or availability, and book lessons with specific horses. These interactions are connected directly to Firebase Firestore, so any updates are reflected in real time for all users.

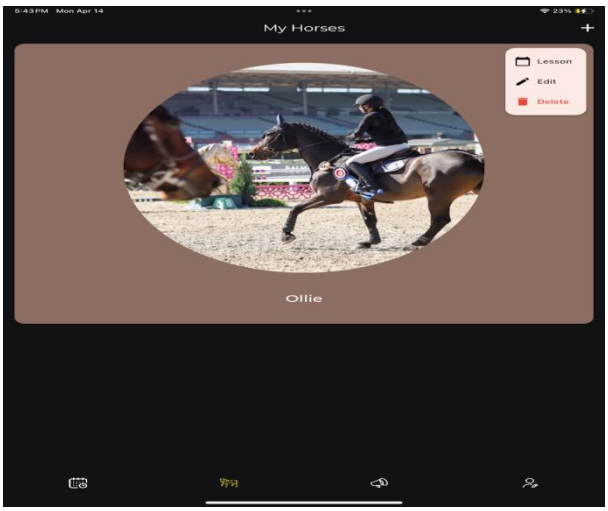
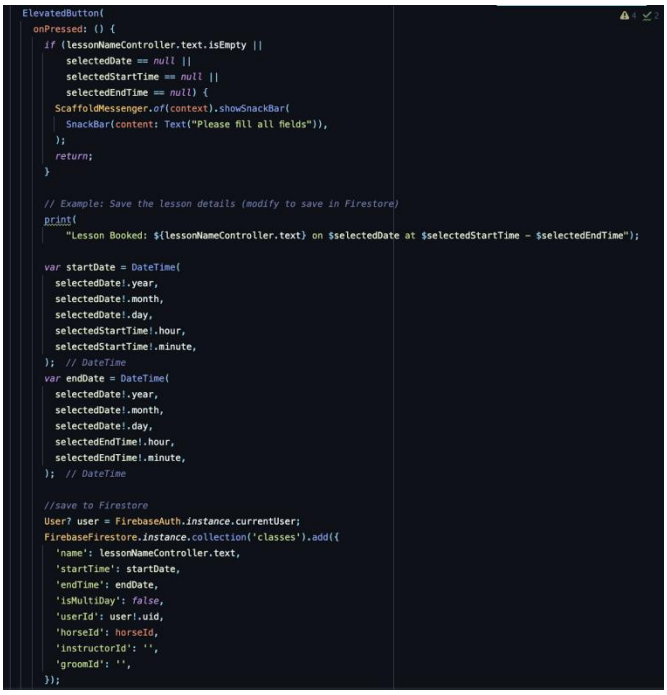


Figure 6. Screenshot of the horses



```
ElevatedButton(
  onPressed: () {
    if (lessonNameController.text.isEmpty ||
        selectedDate == null ||
        selectedStartTime == null ||
        selectedEndTime == null) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Please fill all fields")),
      );
      return;
    }

    // Example: Save the lesson details (modify to save in Firestore)
    print(
        "Lesson Booked: ${lessonNameController.text} on $selectedDate at $selectedStartTime - $selectedEndTime");

    var startDate = DateTime(
      selectedDate!.year,
      selectedDate!.month,
      selectedDate!.day,
      selectedStartTime!.hour,
      selectedStartTime!.minute,
    );  // DateTime
    var endDate = DateTime(
      selectedDate!.year,
      selectedDate!.month,
      selectedDate!.day,
      selectedEndTime!.hour,
      selectedEndTime!.minute,
    );  // DateTime

    //save to Firestore
    User? user = FirebaseAuth.instance.currentUser;
    FirebaseFirestore.instance.collection('classes').add({
      'name': lessonNameController.text,
      'startTime': startDate,
      'endTime': endDate,
      'isMultiDay': false,
      'userId': user!.uid,
      'horseId': horseId,
      'instructorId': '',
      'groomId': '',
    });
```

Figure 7. Screenshot of code 3

This code shows how users could book a lesson through the horse page with HorsiTask. This feature was made with firebase firestore. When the user presses on the button, the app checks if all required fields (lesson names, date, start time, and end time) are filled. If any are missing, it shows a message asking the user to complete the form. Once the inputs are validated, the app creates two “date times” objects, one is for when class starts and one is for when class ends. It then uses Firebase authentication to identify the current user and stores the lesson details in Firestore under “classes” collection. The saved data includes the lesson name and start and end times. This ensures that all lesson bookings are stored securely and can be accessed across the app by trainers, riders and grooms.

4. EXPERIMENT

4.1. Experiment 1

A possible blindspot in HorsiTask is located in the lesson scheduling system. It is important that the app prevents double booking and shows real time updates to avoid scheduling problems between the users, mainly trainers and riders.

To test whether double booking is being prevented, we will simulate a scenario in which multiple users try to book the same time slot with the same trainer. Using test user accounts on Firebase, we will write a script that attempts to book lessons at the same time from different devices. This will help us verify if the system prevents the second booking while allowing only the first. Firestore transaction logs and UI feedback (e.g., booking failure message) will be used as metrics. We will compare this with a control test where booking restrictions are temporarily removed to observe the double-booking issue.

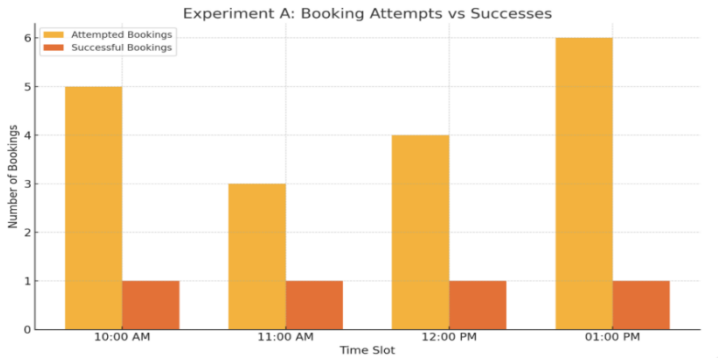


Figure 8. Figure of experiment 1

The experiment showed that despite multiple users attempting to book the same time slot, only one booking was accepted per time. The mean number of attempted bookings per time slot was 4.5, and the median was 4. The lowest attempted was 3, and the highest was 6. The number of successful bookings was always 1, as intended. What was surprising was how quickly the failed booking messages appeared, on average within 2 seconds which indicated that Firestore transactions and snapshot listeners were functioning efficiently. This performance suggests the system works well under stress and maintains data integrity. The biggest factor in the success was Firebase Firestore's ability to perform atomic transactions, which prevents concurrent writes from overlapping. The control group, with booking restrictions disabled, showed duplicate entries for a single time slot, confirming the system's necessity. These results validate the robustness of our scheduling mechanism.

4.2. Experiment 2

Another possible blind spot in HorsiTask is whether the form validation system properly prevents users from submitting incomplete or incorrect data. Ensuring clean data is essential for usability and functionality of the application

To test the form validation system, we created four different input scenarios: missing required fields, entering text in numerical fields, choosing a past date, and submitting a fully valid entry. Using a test environment, we attempted 10 submissions for each scenario. The test measured how many invalid submissions were blocked versus how many were accepted. We also included a control group with all valid inputs to ensure the system allows correct data. This experiment helps determine if the input validation logic works consistently and can prevent common user errors that would otherwise disrupt calendar schedules, lesson accuracy, or horse data integrity.

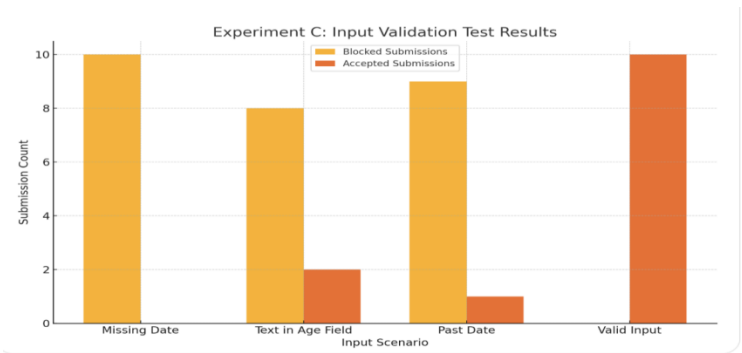


Figure 9. Figure of experiment 2

The form validation system successfully blocked the majority of invalid inputs. For the "Missing Date" scenario, all 10 submissions were blocked. In the "Text in Age Field" case, 8 were blocked, and 2 slipped through. For "Past Date," 9 were blocked and 1 was accepted. Only the valid input group had all 10 accepted. The mean blocked submission rate for invalid inputs was 9, with a median of 9. The lowest was 8, and the highest was 10. These results show that HorsiTask has a robust input validation system, but a few edge cases still allow flawed data. The biggest effect on the result came from form field configuration and the thoroughness of input checks in Flutter's form validation logic. Improvements could be made by adding stricter formatting rules or additional backend checks to catch what the frontend might miss. Still, the validation system overall performs reliably under most typical use cases.

5. RELATED WORK

The article "Mastering the Reins: Key Principles for Horse Barn Management" from Sports Facility Expert explains how good organization helps a barn run smoothly [3]. It talks about keeping up with daily tasks like feeding, cleaning stalls, and grooming, and also making sure schedules for lessons or clinics are clear and easy to follow. The article suggests using simple tools like mobile apps to help with planning, especially if the barn doesn't have a strong internet. This solution works well for staying organized and making sure nothing is forgotten, but it doesn't talk much about how to train staff, build teamwork, or understand horse behavior. What it does well is focus on how to make barn work more efficient and clear, which can help prevent mistakes and save time.

The Colorado State University resource "Horses: Day-to-Day Managers" focuses on giving practical biosecurity guidance to those who care for horses on a daily basis, such as barn managers and staff [4]. Its solution works by breaking down key practices like quarantine procedures, hygiene protocols, and traffic control into simple, actionable steps that can be easily followed in everyday routines. This makes biosecurity more approachable and likely to be used correctly. The approach is effective because it targets the people directly responsible for a horse's daily environment, but it has limitations: it relies on consistent human effort, doesn't deeply address long-term infrastructure challenges, and may overlook less obvious risks. Still, by speaking directly to the day-to-day caretakers, the project improves on more generalized or technical solutions by making careful management a routine part of horse care.
The article "The Importance of Barn Culture" from The Chronicle of the Horse emphasizes how the values, communication, and routines within a barn environment impact the well-being of both horses and people [5]. Its solution focuses on building a positive barn culture through respectful relationships, clear expectations, and consistent practices, which helps create a safe, organized, and enjoyable atmosphere. This approach is effective because it encourages teamwork and accountability, leading to better care and smoother daily operations. However, it depends heavily on strong leadership and buy-in from everyone involved, and it may overlook structural or logistical challenges that good culture alone can't fix. By focusing on daily behavior and shared responsibility, this resource strengthens the human side of horse care, offering an improvement over strategies that focus only on rules or technical procedures.

6. CONCLUSIONS

One major limitation of HorsiTask is its heavy reliance on stable Wi-Fi or internet connection. Because the app uses Firebase for authentication, real-time updates, and data storage, users must be connected to the internet to access most features [13]. In barns located in rural areas or places with poor network coverage, this could result in delayed updates, failed lesson bookings, inability to load schedules and horse information, or issues login in and signing up

into the application. This reliance on Wi-Fi may delay the app's effectiveness in environments where connectivity is unreliable. If given more time, I would implement offline caching features that allow users to view and modify their schedules, horse data, and lesson requests without needing constant internet access. Changes made offline could then automatically sync with Firebase when a connection is reestablished. This improvement would make HorsiTask much more flexible and usable across a wider variety of barn environments.

HorsiTask offers a modern solution to barn management challenges by streamlining scheduling, communication, and horse care. Although its reliance on a stable internet presents limitations, the app's overall design enhances organization and efficiency. With future improvements like offline functionality, HorsiTask can become an essential tool for barns of all sizes.

REFERENCES

[1] Forrest, Susanna. The age of the horse: An equine journey through human history. Open Road+ Grove/Atlantic, 2017..
[2] Rosman, Amirul Asyraf, and Hadafi Fitri Mohd Latip. "Mobile Application for Horse Training Program System." Journal of Human Centered Technology 1.1 (2022): 48-53.
[3] Brown, Jeremy Houghton, Sarah Pilliner, and Zoe Davies. Horse and stable management. John Wiley & Sons, 2013.
[4] Tokach, M. D., et al. "Feeding and barn management strategies that maximize feed efficiency." Feed efficiency in swine. Wageningen Academic, 2012. 41-62.
[5] Lawson, R. "Barn raising: Fostering a collaborative culture to design curriculum in HE." 14th International Conference on Knowledge, Culture and Change in Organizations and the Organization knowledge community. 2014.
[6] Huntington, Peter, Jane Myers, and Elizabeth Owens. Horse Sense: the guide to horse care in Australia and New Zealand. Landlinks Press, 2004.
[7] Swapna, S., et al. "Data cleaning for data quality." 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2016.
[8] Moroney, Laurence. "Using authentication in firebase." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 25-50.
[9] Kim, Jae-Jung, and Seng-Phil Hong. "A method of risk assessment for multi-factor authentication." Journal of Information Processing Systems 7.1 (2011): 187-198.
[10] Polyzotis, Neoklis, et al. "Data validation for machine learning." Proceedings of machine learning and systems 1 (2019): 334-347.
[11] Pramono, Luthfan Hadi, and Yohanes Krisna Yana Javista. "Firebase authentication cloud service for RESTful API security on employee presence system." 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). IEEE, 2021.
[12] Luke, Karen L., et al. "New insights into ridden horse behaviour, horse welfare and horse-related safety." Applied Animal Behaviour Science 246 (2022): 105539.
[13] Siddiqa, Aisha, Ahmad Karim, and Abdullah Gani. "Big data storage technologies: a survey." Frontiers of Information Technology & Electronic Engineering 18 (2017): 1040-1070.
[14] Ullah, Shahid, et al. "An innovative user-friendly platform for Covid-19 pandemic databases and resources." Computer Methods and Programs in Biomedicine Update 1 (2021): 100031.
[15] Kesavan, Ram, et al. "Firestore: The nosql serverless database for the application developer." 2023 IEEE 39th International Conference on Data Engineering (ICDE). IEEE, 2023.