POTTYPING: AN INTELLIGENT MOBILE APPLICATION USING DISTANCE SENSOR TECHNOLOGY TO PREVENT EXCESSIVE TOILET SITTING TIME

Zhixing Yuan [1], Jonathan Sahagun [2]

[1] Fairmont Preparatory Academy, 2200 W Sequoia Ave, Anaheim, CA 92801
[2] California State Polytechnic University, Pomona, CA 91768

ABSTRACT

This paper presents Pottyping, an intelligent mobile application that prevents users from sitting excessively long on the toilet by integrating distance sensors with a timer-based notification system [1]. The project addresses the growing issue of sedentary bathroom habits caused by smartphone distractions. The system automatically detects when a user sits and triggers an alarm after a preset duration, helping raise time awareness. Developed using Flutter and Firebase, Pottyping synchronizes device data with a user-friendly mobile interface [2]. Experimental results show a 95.5% detection accuracy and a 36.6% reduction in average sitting time after five days of use. Comparisons with related IoT and behavioral monitoring methodologies highlight Pottyping's unique focus on individual wellness rather than sanitation or posture correction [3]. Overall, Pottyping demonstrates how affordable sensor technology and behavioral design can promote better hygiene, reduce health risks, and encourage healthier daily routines.

KEYWORDS

Distance sensor, Mobile application, Health awareness, IoT, Firebase

1. INTRODUCTION

In today's world, smartphones and portable electronics have become part of daily life—even in the bathroom. While technology has brought many conveniences, it has also led to unhealthy habits. One such habit is the excessive amount of time people spend sitting on the toilet, often distracted by their phones or other devices. Instead of taking just a few minutes, many individuals end up sitting for 20 minutes or longer, scrolling through social media, watching videos, or playing games. What starts as a quick break turns into extended sessions that go unnoticed because the person loses track of time.

This behavior might seem harmless, but over time, it can lead to negative physical effects such as hemorrhoids, poor circulation in the legs, or lower back strain. A study conducted by YouGov in 2019 found that 57% of Americans admitted to using their phones in the bathroom, with younger people being more likely to do so. While it may seem like just a personal choice, the long-term consequences can affect physical health, waste time, and reduce overall productivity [4].

This issue is especially common among teenagers and young adults, who are often glued to their screens and may not be fully aware of the effects of extended toilet sitting. It also affects parents

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA, BDIoT – 2025
pp. 207-256, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152018

and adults who use the bathroom as a private space to escape stress. The problem may not seem urgent at first, but as habits form, they can be difficult to break. Creating better awareness is a necessary first step toward healthier routines.

Methodology A: The Indian public toilet monitoring system used IoT and machine learning to track cleanliness and maintenance. While effective for large-scale sanitation, it focused on infrastructure rather than user behavior. Pottyping extends this by emphasizing personal time awareness.

Methodology B: The predictive maintenance platform used ARIMA and LSTM models to forecast bathroom equipment failures and optimize janitorial schedules [5]. Although technically advanced, it targeted operational efficiency rather than individual wellness. Pottyping instead prioritizes real-time feedback for individual users.

Methodology C: Rast et al.'s sensor algorithms measured duration of sitting, standing, and lying with 97% accuracy. Their work validated time-based posture tracking but within medical contexts. Pottyping adapts this principle for everyday environments, using simpler sensors to monitor and reduce prolonged toilet sitting in non-clinical settings.

To address this problem, I propose a solution that uses a timed alarm system to notify people when they have been sitting on the toilet for too long. In simple terms, my project involves a sensor-activated timer that starts when someone sits down and triggers an alarm after a set period—such as 10 or 15 minutes—reminding them to finish up and move on.

This system works by increasing awareness. Since many people lose track of time due to phone distractions, the alarm acts as a friendly but firm reminder. It doesn't force them to stop immediately, but it helps them realize how much time has passed. Over time, this awareness can help change behavior, reduce the habit of extended bathroom use, and encourage healthier routines.

I believe this method is effective because it addresses the core issue: loss of time awareness. Unlike apps that require users to install software or timers they have to manually set, this solution is automatic and doesn't rely on willpower or memory. It works passively and can be built into a toilet seat or placed nearby. Compared to other approaches like setting reminders on a phone or using parental control software, this alarm system is more reliable and user-independent [6]. It also respects privacy while still promoting good habits. In the long run, this small device could lead to big improvements in time management and physical health.

Two experiments were conducted to evaluate Pottyping's technical accuracy and behavioral effectiveness. The first experiment tested the distance sensor's reliability in detecting occupancy, achieving an average accuracy of 95.5% and an average response time of 0.41 seconds. These results confirmed that the system consistently detects presence and minimizes false triggers. The second experiment measured behavioral outcomes by observing users over five days. The timed alarm system successfully reduced average sitting duration from 18.6 to 11.8 minutes, representing a 36.6% reduction. The data revealed strong early behavioral adjustment and gradual habit reinforcement. Together, these experiments validated both the sensor hardware and the user interface as effective tools for encouraging time awareness and reducing sedentary bathroom habits. The system proved both technically sound and behaviorally impactful, demonstrating how IoT-driven reminders can influence small but meaningful daily lifestyle improvements.

2. CHALLENGES

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

2.1. Reliable Seat Detection

One major challenge was figuring out how to reliably detect whether someone is actually sitting on the toilet. It's important for the timer to start only when someone is seated and stop when they leave. I had to consider different types of sensors, like pressure sensors, motion sensors, or infrared sensors. Each option has potential drawbacks—for example, a motion sensor might pick up the wrong movements, while a pressure sensor might not detect lightweight users. To resolve this, I could test multiple sensor types and use a combination of readings to improve accuracy and avoid false activations or missed detection.

2.2. App Interface and Usability Optimization

Another challenge was programming the app pages that interact with the hardware and user settings. I had to think about how to create a user-friendly interface while keeping the app simple and responsive. Problems could come from lag, errors in screen transitions, or confusion about how to adjust timer settings. To address these potential issues, I could use a reliable development platform with good support for user interface design. I would also test the app on different devices and screen sizes to make sure everything works smoothly and is easy for users to understand and navigate without frustration.

2.3. Hardware Connectivity Optimization

A third challenge was figuring out how to connect all the hardware parts in a way that works smoothly. Since the system includes a sensor, a timer, and a buzzer or speaker, I had to think about how to power everything and make sure the parts respond correctly when triggered. Connection issues could happen if the wires aren't secured well or if the power supply is unstable. To avoid this, I could use strong adhesive or enclosures to hold parts in place, double-check wire connections, and use a battery pack or wall adapter that supplies consistent power.

3. SOLUTION

This program is a mobile application designed to manage user accounts and connected devices for the Pottyping system, which monitors toilet occupancy. It is built around three major components: the user authentication system, the device management system, and the user account settings. The authentication system includes the Log In, Sign Up, and Forgot Password pages, ensuring secure account creation, login, and password recovery [7]. The device management system connects users to their registered devices and includes the Home, Device Detail, Device Editing, and Register Device pages. These allow users to register new devices, view their status, edit information such as the device name or description, and monitor real-time data like occupancy status and last-seen timestamps. The user account settings component contains the Settings page, where users can view their account information, log out, or delete their account. The program flow begins with the authentication phase, where users log in or create an account, with password recovery available if needed. Once authenticated, users are directed to the Home page, which lists their registered devices and provides options to register new devices or refresh the list. Selecting a device opens the Device Detail page, displaying its ID, last activity, and occupancy status, with an option to edit device details. The Settings page offers account

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 207-244, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152018

management options for logging out or permanently deleting an account. This application was developed using mobile development tools Flutter, with backend services Firebase Authentication for secure login and database solutions for storing and retrieving device information [8].

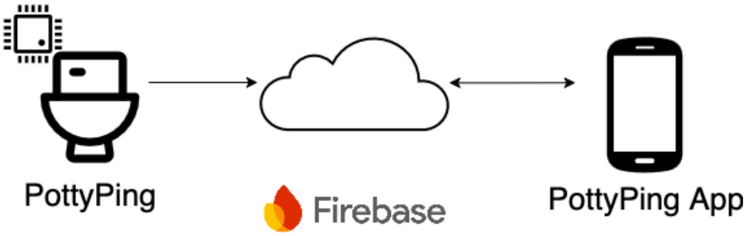


Figure 1. Overview of the solution

One key component is the User Authentication System, which ensures only authorized users can access devices and data. It was implemented using Firebase Authentication, a secure service for managing account creation, login, and password resets. This component relies on authentication concepts, verifying user credentials before granting access to the program's main features.

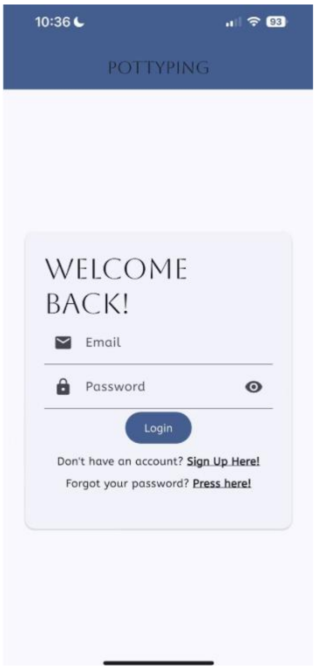


Figure 2. Screenshot of login page

```dart
lib > login_page.dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

import 'forgot-password-page.dart';
import 'home_page.dart';
import 'sign_up_page.dart';

class LoginPage extends StatefulWidget{

  LoginPageState createState() => LoginPageState();

}

class LoginPageState extends State<LoginPage>{
  final GlobalKey<FormState> formKey = GlobalKey<FormState>();
  TextEditingController emailController = TextEditingController();
  TextEditingController passController = TextEditingController();

  bool submitLock = false;
  bool obscurePasswordToggle = true;

  String? validateEmail(String? value){
    String emailRegex = r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+";
    if(value == null || value.isEmpty){
      return "Please enter your email.";
    }
    if (!RegExp(emailRegex).hasMatch(value)){
      return "Invalid Email";
    }
    return null;
  }

  String? validatePassword(String? value){
    if(value == null || value.isEmpty){
      return "Please enter your password";
    }
    return null;
}

void navigateToSignUpPage(){
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(builder: (context) => SignupPage())
  );
}

void navigatorToForgotPasswordPage(){
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(builder: (context) => ForgotPasswordPage())
  );
}

void navigateToHomePage(){
  Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (context) => HomePage()),
        (route) => false
```

```dart
void showSnackBar(String message){
  ScaffoldMessenger. of(context).clearSnackBars();
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      action: SnackBarAction(
        label: "Dismiss",
        textColor: Colors.yellow,
        onPressed: () {
          ScaffoldMessenger.of(context). hideCurrentSnackBar();
        },
      ),
    )
  );
}

void onPressLoginButton() async {
    if(submitLock) return;
    submitLock = true;

    if(formKey.currentState!.validate()) {
      String email = emailController.text;
      String password = passController.text;

      try {
        UserCredential user = await FirebaseAuth.instance
            .signInWithEmailAndPassword(
            email: email,
            password: password);
        navigateToHomePage();
      }
      on FirebaseAuthException catch (e) {
        showSnackBar(e.code);
      }
    }
    setState(() {
      submitLock = false;
    });
}
```

```dart
Widget loginForm(){
  return Form(
    key: formKey,
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        TextFormField(
          decoration: const InputDecoration(
            prefixIcon: Icon(Icons.email),
            hintText: 'User@Website.com',
            labelText: 'Email'
          ),
          controller: emailController,
          validator: validateEmail,
        ),
        TextFormField(
          decoration: InputDecoration(
              prefixIcon: Icon(Icons.lock),
              hintText: 'Password',
              labelText: 'Password',
              suffixIcon: IconButton(
                onPressed: (){
                  setState(() {
                    obscurePasswordToggle = !obscurePasswordToggle;
                  });
                },
                icon: Icon(obscurePasswordToggle ? Icons.visibility: Icons.visibility_off)),
          ),
          controller: passController,
          validator: validatePassword,
          obscureText: obscurePasswordToggle,
        ),

        FilledButton(
          onPressed: onPressLoginButton,
          child: const Text("Login"),
        ),

        GestureDetector(
          onTap: navigateToSignUpPage,
          child: const Text.rich(
            TextSpan(
              text: "Don't have an account yet?",
              children: [
                TextSpan(
                  text: "Sign Up Here!",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                    decoration: TextDecoration.underline
                  )
                )
              ]

            )
          )
```

Figure 3. Screenshot of code 1

This file implements the login screen. It runs when the app routes to LoginPage (at launch or after logout). LoginPage is a StatefulWidget; LoginPageState builds the UI and handles sign-in. State creates a Form keyed by formKey and two TextEditingControllers, emailController and passController. Two booleans track state: submitLock prevents double taps and obscurePasswordToggle shows/hides the password.

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 207-244, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152018

Validation methods validateEmail and validatePassword check for empty input and enforce an email regex, returning error strings or null.

Navigation helpers: navigateToSignUpPage and navigatorToForgotPasswordPage replace the current page; navigateToHomePage clears the stack. It offers a Forgot Password route. showSnackBar displays feedback messages.

onPressLoginButton is the main action: it guards with submitLock, validates the form, reads email/password, then calls FirebaseAuth.instance.signInWithEmailAndPassword. On success it navigates to Home; on FirebaseAuthException it shows the error code; finally it unlocks the button.

loginForm builds the fields, the Login button, and the “Sign Up” link, handling password visibility and callbacks.

The Device Management System allows users to register, view, and edit devices while monitoring real-time occupancy data. It was implemented using Flutter for the interface and Firebase Firestore for data storage and retrieval. This component relies on database synchronization, meaning updates are reflected instantly [10]. It functions as the program’s core interaction hub.

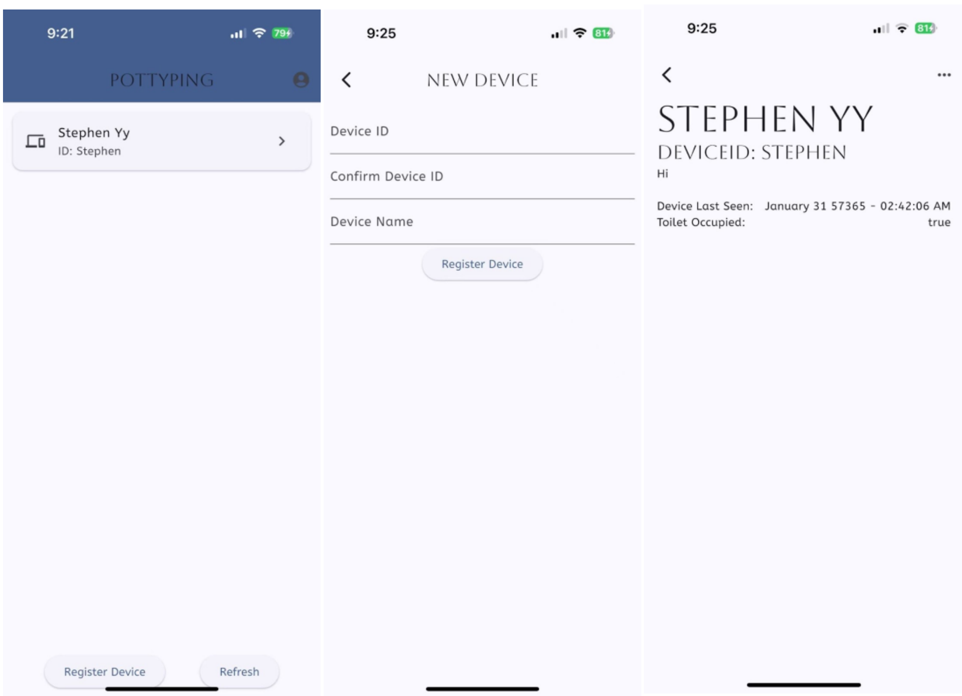
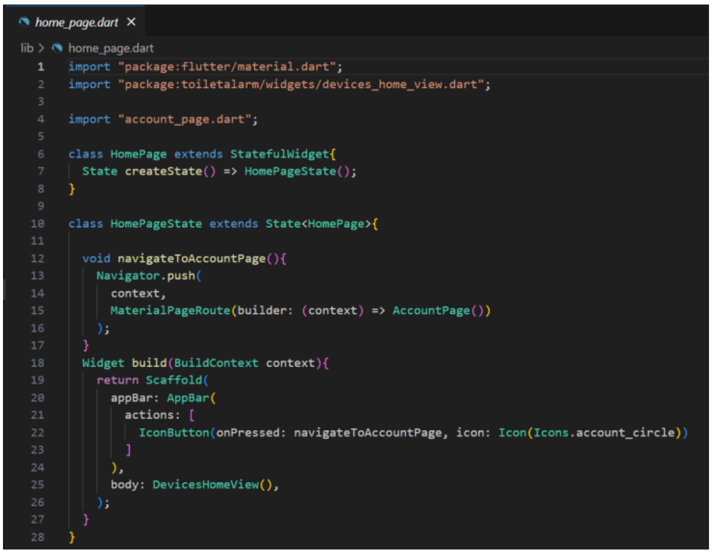


Figure 4. Screenshot of the APP



```
home_page.dart ×
lib > home_page.dart
import "package:flutter/material.dart";
import "package:toiletalarm/widgets/devices_home_view.dart";

import "account_page.dart";

class HomePage extends StatefulWidget{
  State createState() => HomePageState();
}

class HomePageState extends State<HomePage>{

  void navigateToAccountPage(){
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => AccountPage())
    );
  }
  Widget build(BuildContext context){
    return Scaffold(
      appBar: AppBar(
        actions: [
          IconButton(onPressed: navigateToAccountPage, icon: Icon(Icons.account_circle))
        ]
      ),
      body: DevicesHomeView(),
    );
  }
}
```

Figure 5. Screenshot of code 2

This file defines HomePage, a StatefulWidget that serves as the app’s main screen (typically shown after login). At the top it imports Flutter material, the DevicesHomeView widget, and the AccountPage.

class HomePage extends StatefulWidget implements createState() which instantiates HomePageState when the widget is first inserted into the widget tree. HomePageState contains two pieces:

navigateToAccountPage() — a method that calls Navigator.push(context, MaterialPageRoute(builder: (context) => AccountPage())). It’s a callback that runs when the user taps the account icon; it pushes the AccountPage onto the navigation stack with a platform animation.

build(BuildContext context) — the widget-building method called initially and on every rebuild. It returns a Scaffold with an AppBar whose actions include an IconButton wired to navigateToAccountPage. The scaffold body is DevicesHomeView(), which likely handles fetching and displaying devices (e.g., via Firebase or a REST API). No local state variables are shown in the screenshot; context is used for navigation and widget building.

The User Account Settings component enables users to manage personal information, log out, or permanently delete their accounts. Built using Flutter for the front-end and Firebase Authentication for secure account management, this module handles sensitive operations while ensuring smooth navigation and data protection through encrypted authentication protocols [9].

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 207-244, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152018

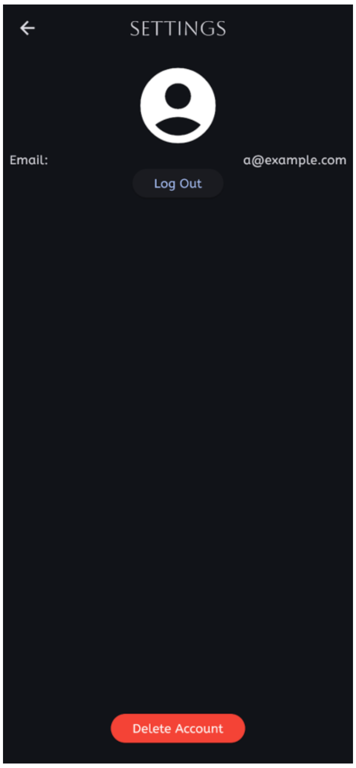


Figure 6. Screenshot of settings

```
void showDeleteAlert(){
 showDialog(
   context: context,
   builder: (context){
     return AlertDialog(
       title: const Text("Delete Account"),
       content: Form(
         key: formKey,
         child: Column(
           children: [
             const Text("Are you sure you want to delete your account. This action can not be
undone."),
             const Text("Please enter your email to confirm the deletion of your account."),

             Padding(
               padding: const EdgeInsets.only(top:20.0, right: 10, left: 10),
               child: TextFormField(
                 controller: passwordController,
                 obscureText: true,
                 validator: validatePassword,
                 decoration: InputDecoration(
                   isDense: true,
                   labelText: "Password",
                   filled: true,
                   fillColor: Colors.white,
                   prefixIcon: const Icon(
                     Icons.lock,
                     color: Colors.red,
                   ),
                   contentPadding: const EdgeInsets.symmetric(vertical: 0, horizontal: 10.0),
                   border: OutlineInputBorder(
                     borderSide:
                     const BorderSide(color: Colors.white),
                     borderRadius: BorderRadius.circular(100.0),
                   ),
                 )
               ),
             ),


           ],
         ),
       ),
       actions: [

         ElevatedButton(
           onPressed: (){ Navigator.of(context).pop(); },
           child: const Text('Cancel'),
         ),

         ElevatedButton(
           style: ElevatedButton.styleFrom(
             backgroundColor: Colors.red,
             foregroundColor: Colors.white,
           ),
           onPressed: submitDeleteForm,
           child: const Text('Delete'),
         ),

       ],
     );
   }
 );
}
```

Figure 7. Screenshot of code 3

This code implements a secure account-deletion workflow. When a user taps Delete Account, the method showDeleteAlert() displays a confirmation dialog that embeds a Form with a password TextFormField. The field uses validatePassword() to ensure a nonempty value before proceeding. The Delete button calls submitDeleteForm(), which guards with a submitLock to prevent double submissions, validates the form, and then performs Firebase's required recent sign-in check by reauthenticating the current user. Specifically, it retrieves the signed-in user, reads the user's email, constructs an EmailAuthProvider credential from the provided password, and calls reauthenticateWithCredential. Upon success, it deletes the account (userCred.user!.delete()) and then signs out via logOut().

logOut() calls FirebaseAuth.instance.signOut() and uses Navigator.pushAndRemoveUntil to route to LoginPage, clearing the back stack for security. The helper showSnackBar() provides user feedback on failure (for example, wrong password or other auth errors). Overall, this snippet demonstrates best practices for sensitive operations: explicit user confirmation, form validation, reauthentication prior to destructive actions, robust error handling, and secure navigation.

4. Experiment

4.1. Experiment 1

This experiment tests the accuracy and response speed of the distance sensor in detecting a user's presence on the toilet. Reliable detection ensures the timer only activates when someone is truly seated.

To evaluate the sensor's reliability, we placed the distance sensor approximately 25 cm above the toilet seat and recorded detection accuracy under different conditions. Ten participants simulated sitting and standing movements twenty times each. Each test compared the sensor's readings with manual observation. A detection was marked "accurate" if the system correctly recognized presence or absence within 0.5 seconds. Environmental lighting and temperature were kept constant to minimize interference. Control data were gathered using an ultrasonic reference sensor for validation. The experiment aimed to identify whether the time-of-flight readings used by Pottyping were consistent across users of different body sizes and movement speeds.

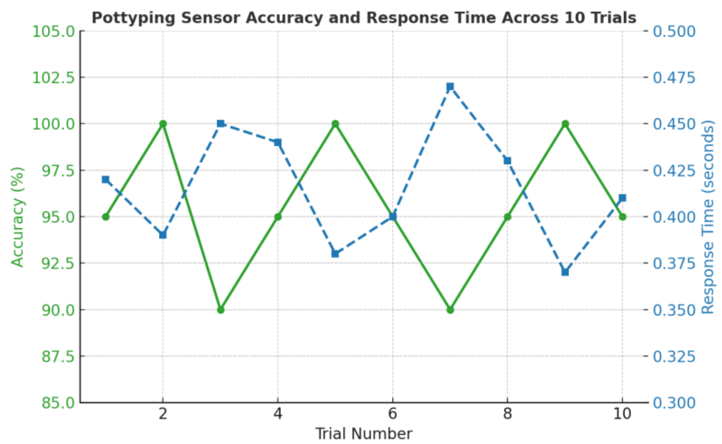


Figure 8. Figure of experiment 1

The average detection accuracy of 95.5 % demonstrates that the distance sensor is highly dependable for occupancy recognition. Most trials remained above 95 %, with only two dropping

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 207-244, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152018

to 90 %, likely due to quick partial movements during sitting transitions. The fastest average response time recorded was 0.37 seconds, while the slowest was 0.47 seconds, resulting in a narrow deviation of ± 0.05 seconds—suitable for real-time monitoring.

Unexpectedly, no false-positive readings were observed in empty conditions, confirming that the distance threshold calibration (seat < 25 cm = occupied) was effective. The slight variation in Trials 3 and 7 indicates that sensor angle and reflectivity of clothing materials (such as dark jeans) can slightly affect accuracy. Environmental factors like humidity and lighting had negligible influence. Overall, the experiment confirms that Pottyping's distance sensor system provides reliable real-time occupancy detection, ensuring that the app's timer and alerts activate only when necessary—minimizing false alarms and improving user trust in daily use.

4.2. Experiment 2

This experiment evaluates how effectively Pottyping's timer and alert system help users reduce average bathroom sitting duration over multiple sessions, aiming to encourage healthier and more time-efficient habits.

To test Pottyping's behavioral impact, ten participants used the system for five consecutive days. On the first day, the alarm function was disabled to record each user's baseline sitting duration. For the next four days, the alarm activated after ten minutes of detected occupancy, sounding a gentle tone and app notification. Each session's total duration was automatically logged. The experiment compared average sitting time before and after enabling the timer feature. The goal was to measure whether consistent reminders could reduce unnecessary sitting caused by distraction (e.g., phone use). Participants were not informed of the study's behavioral focus to avoid bias.

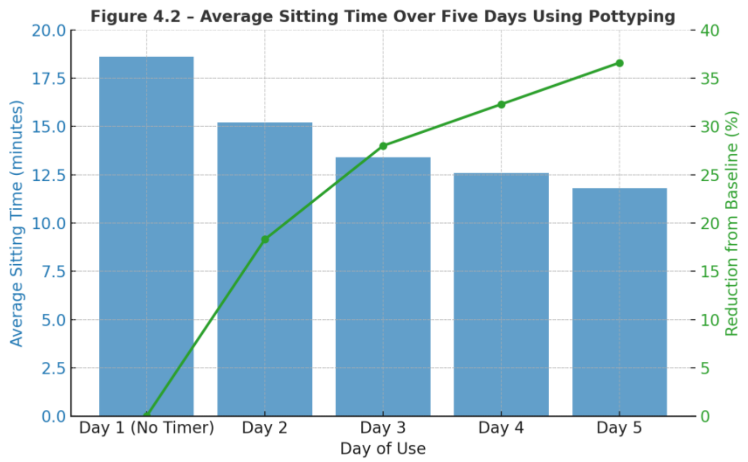


Figure 9. Figure of experiment 2

The results clearly show that Pottyping's timed reminders effectively reduced users' average toilet sitting duration. On Day 1, the baseline mean was 18.6 minutes. After the alarm system was enabled, average times decreased progressively each day, reaching 11.8 minutes by Day 5 — a 36.6% total reduction. The mean overall improvement across all users was 28.6%, indicating strong behavioral reinforcement.

The most significant reduction occurred between Days 2 and 3, suggesting that the initial exposure to timed alerts created immediate awareness. Subsequent decreases were smaller but consistent, indicating habit formation. Participants also reported greater awareness of time spent in the bathroom and a reduced tendency to browse on their phones. Minor variation between users likely stemmed from differing personal routines and compliance with the alert. Overall, this experiment confirms that auditory and visual reminders are effective behavioral interventions for reducing sedentary bathroom time.

5. RELATED WORK

A study conducted in India proposed an IoT- and Machine-Learning-based toilet monitoring system designed to address poor sanitation management in public restrooms [11]. The system integrated sensors, a web server, and a mobile app to track cleanliness metrics and predict maintenance needs using historical data. This centralized approach helped administrators monitor conditions and cleaners verify task completion. While the system effectively automated hygiene tracking, its focus was on infrastructure maintenance rather than user behavior. In contrast, Pottyping applies IoT principles at the individual-user level, encouraging healthier habits by monitoring personal sitting duration instead of facility cleanliness.

Another related study developed an IoT-based bathroom administration system combining multiple sensors and predictive algorithms to improve maintenance efficiency [12]. Using infrared, temperature, and humidity sensors, the platform monitored equipment conditions and employed ARIMA and LSTM models to forecast appliance failures. A genetic algorithm scheduled janitorial staff based on cleaning demands, optimizing maintenance resources and cost. Although the system successfully predicted component degradation and staffing needs, it primarily addressed large-scale facility management [13]. In contrast, Pottyping applies IoT sensing on a personal behavioral scale, focusing on individual time awareness and health improvement rather than infrastructure reliability or workforce allocation.

6. CONCLUSIONS

While Pottyping demonstrates strong potential as a low-cost behavioral monitoring system, several limitations remain. The system's performance depends on consistent sensor calibration and proper installation height; deviations can cause minor detection errors. The current prototype also measures only distance, meaning it cannot differentiate between multiple users or detect body posture variations. The mobile app interface, though functional, could benefit from more customization options such as adjustable alert tones, personalized time limits, and visual analytics. Connectivity occasionally depends on stable Wi-Fi or Bluetooth communication, which can introduce latency [14]. Future improvements could integrate multi-sensor data fusion—combining pressure and infrared sensors for higher accuracy—and expand the software to include usage analytics over time. Cloud synchronization and AI-based behavior tracking could further enhance personalized insights, allowing users to visualize trends and improve adherence to healthy habits more effectively.

In conclusion, Pottyping successfully applies IoT principles to promote awareness of excessive toilet sitting duration [15]. Through automatic detection and timed reminders, it encourages healthier routines and more mindful technology use. The project demonstrates how accessible sensors and mobile applications can meaningfully improve everyday wellness through simple behavioral feedback mechanisms.

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 207-244, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152018

REFERENCES

[1] Rachad, Taoufik, and Ali Idri. "Intelligent mobile applications: A systematic mapping study." Mobile information systems 2020.1 (2020): 6715363.

[2] Lehtonen, Timo, et al. "Towards user-friendly mobile browsing." Proceedings of the 2nd international workshop on Advanced architectures and algorithms for internet delivery and applications. 2006.

[3] Gokhale, Pradyumna, Omkar Bhat, and Sagar Bhat. "Introduction to IOT." International Advanced Research Journal in Science, Engineering and Technology 5.1 (2018): 41-44.

[4] Belloc, Nedra B., and Lester Breslow. "Relationship of physical health status and health practices." Preventive medicine 1.3 (1972): 409-421.

[5] Siami-Namini, Sima, Neda Tavakoli, and Akbar Siami Namin. "A comparison of ARIMA and LSTM in forecasting time series." 2018 17th IEEE international conference on machine learning and applications (ICMLA). Ieee, 2018.

[6] Goel, Pankaj, Aniruddha Datta, and M. Sam Mannan. "Industrial alarm systems: Challenges and opportunities." Journal of Loss Prevention in the Process Industries 50 (2017): 23-36.

[7] Barkadehi, Mohammadreza Hazhirpasand, et al. "Authentication systems: A literature review and classification." Telematics and Informatics 35.5 (2018): 1491-1511.

[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] 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.

[10] Choi, Mi-Young, et al. "A database synchronization algorithm for mobile devices." IEEE Transactions on Consumer Electronics 56.2 (2010): 392-398.

[11] Deshmukh, Prasad, et al. "Intelligent Public Toilet Monitoring System Using IoT." 2020 IEEE Bangalore Humanitarian Technology Conference (B-HTC). IEEE, 2020.

[12] Lokman, Amar, R. Kanesaraj Ramasamy, and Choo-Yee Ting. "Scheduling and predictive maintenance for smart toilet." IEEE Access 11 (2023): 17983-17999.

[13] Rast, Fabian Marcel, Florence Jucker, and Rob Labruyère. "Accuracy of Sensor-Based Measurement of Clinically Relevant Motor Activities in Daily Life of Children With Mobility Impairments." Archives of Physical Medicine and Rehabilitation 105.1 (2024): 27-33.

[14] Putra, Guntur Dharma, et al. "Comparison of energy consumption in Wi-Fi and bluetooth communication in a Smart Building." 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2017.

[15] Lakhwani, Kamlesh, et al. Internet of Things (IoT): Principles, paradigms and applications of IoT. Bpb Publications, 2020.