

ระบบ จัดการห้องพัก (RoomRover)
Dormitory Management System (RoomRover)

เสนอ
อาจารย์ภัทร อัยรักษ์

นำเสนอโดย

6410210102 นายณัฐภาส ขำเกิด

6410210776 นางสาวฮัซมา อุซัยง

6510210073 นายชินพงศ์ รักใหม่

6510210107 นายติณห์ ฤทธิไธ

6510210207 นายเปศล ชุ่มจำรัส

สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่

ที่มาและความสำคัญ

ในยุคสมัยใหม่ที่ระบบดิจิทัล การสื่อสารและการทำธุรกิจเปลี่ยนแปลงอย่างรวดเร็ว ความสะดวกสบายและความคล่องตัวในการแลกเปลี่ยนข้อมูลติดต่อจึงกลายเป็นสิ่งสำคัญ ซึ่งการจัดการหอพักก็เป็นส่วนหนึ่งที่สมควรจะเปลี่ยนไป เช่น การจัดการค่าใช้จ่าย การแจ้งการซ่อมบำรุงรักษาที่ลูกค้าแจ้งหอพัก ที่โดยรูปแบบเดิมจะเป็นการแจ้งฝ่ายดูแลหอพักโดยการพูดคุยกันหรือการเขียนลงในกระดาษมอบให้ที่ละห้องซึ่งอาจมีข้อผิดพลาดระหว่างการสื่อสารหรือแม้กระทั่งความไม่เข้าใจซึ่งอาจนำไปสู่ปัญหามากมายในภายหลัง

ระบบจัดการหอพักจึงได้ถูกพัฒนาขึ้นเพื่อแก้ปัญหาดังกล่าว โดยการนำเอาเทคโนโลยีดิจิทัลมาใช้ในการจัดเก็บข้อมูลรวมถึงติดต่อสื่อสารกับลูกค้าโดยยังสามารถอัปเดตข้อมูลได้อย่างรวดเร็ว โดยการออกแบบและสร้างแอปพลิเคชัน

ระบบจัดการหอพักจึงเป็นเครื่องมือที่มีความสำคัญในการทำธุรกิจหอพัก ที่ไม่เพียงช่วยเพิ่มความความสะดวกสบายในการแลกเปลี่ยนข้อมูลติดต่อ แต่ยังสอดคล้องกับเทรนด์ดิจิทัลและความต้องการของผู้ใช้ในยุคใหม่

วัตถุประสงค์ของโครงการ

1. เพื่อให้ฝ่ายดูแลหอพักสามารถจัดการดูแลหอพักได้สะดวกขึ้น
2. ฝ่ายดูแลหอพักสามารถนำข้อมูลไปใช้บริหารงาน ได้อย่างมีประสิทธิภาพมากยิ่งขึ้น
3. เพื่อให้ลูกค้าที่พักอาศัยสามารถบริหารการใช้ไฟ/น้ำ ได้อย่างมีประสิทธิภาพ
4. เพื่อให้ลูกค้าที่พัก สามารถจ่ายค่าที่พักได้สะดวกและมีรายละเอียดชัดเจนยิ่งขึ้น

ขอบเขต

ขอบเขตของโปรเจกต์นี้จะเน้นไปที่การพัฒนาระบบที่สามารถทำให้ฝ่ายดูแลหอพักสามารถจัดการหอพัก ค่าใช้จ่าย รวมถึงการแจ้งซ่อม และทำให้ลูกค้าที่พักสามารถ บริหารค่าใช้จ่ายทั้งค่าน้ำค่าไฟ และการจ่ายเงินที่สะดวกยิ่งขึ้น

ขั้นตอนการดำเนินโครงการ

โครงการนี้ดำเนินงานโดยใช้วงจรการพัฒนาแบบ System Development Life Cycle (SDLC) โดยมีรายละเอียดดังแสดงในตารางที่ 1.1

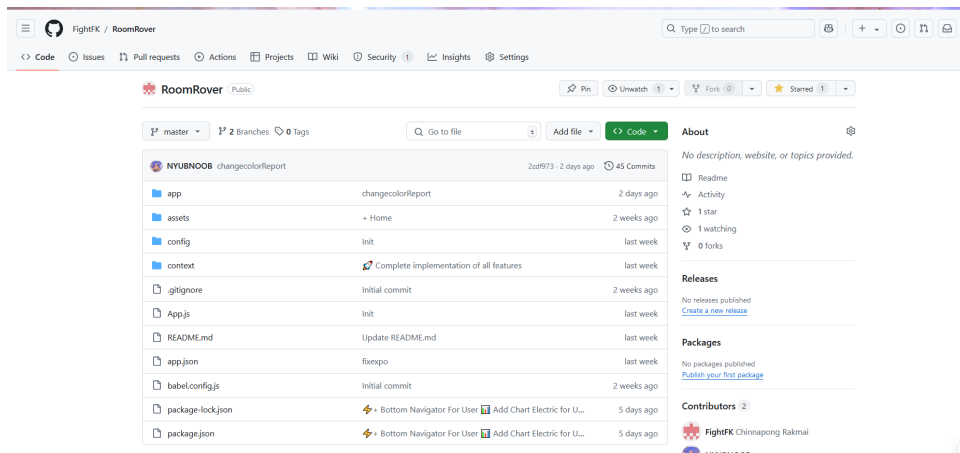
ตารางที่ 1.1 ขั้นตอนการดำเนินงาน

ขั้นที่	กิจกรรม	สิ่งที่ได้
1	กำหนดขอบเขตและวางแผนโครงการ (Project Planning) - ศึกษาปัญหาของนามบัตรแบบกระดาษ - ศึกษาเครื่องมือและเทคโนโลยีที่จะนำมาใช้แก้ไขปัญหาของนามบัตรแบบกระดาษ	- Proposal
2	การวิเคราะห์ระบบ (Analysis) - วิเคราะห์ความต้องการของผู้ใช้งานแอปพลิเคชัน - กำหนดขอบเขตความต้องการของแอปพลิเคชัน	- Use Case Diagram - Data Flow Diagram - E-R Diagram
3	การออกแบบระบบ (Design) - ออกแบบโครงสร้างเมนู - ออกแบบหน้าจอการแสดงผล	- โครงสร้างระบบ
4	การพัฒนาแบบ (Development) - พัฒนาโปรแกรมจากข้อมูลที่ได้วิเคราะห์และออกแบบไว้	- ชุดคำสั่งโปรแกรม
5	การทดสอบและปรับปรุงแก้ไข (Testing) - ทดสอบการใช้งาน - แก้ไขข้อผิดพลาดที่เกิดขึ้น	- ระบบที่สมบูรณ์

วิธีการพัฒนาแอปพลิเคชัน

วิธีการพัฒนาตามสามารถ ดูได้ใน Readme ของ Github และสามารถ Download ไฟล์ของ Project มาเพื่อติดตั้งและลองใช้งานได้ได้ผ่านลิงค์ด้านล่าง รายละเอียดทั้งหมดถูกใส่ไว้ใน Readme

Link To GitHub



เครื่องมือในการพัฒนา

- React Expo ใช้ในการพัฒนาทางฝั่ง frontend
- Firebase ใช้ในการพัฒนาทางฝั่ง backend

Code ในส่วนที่สำคัญ

1. Part App.js

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, View } from 'react-native';
import Home from './app/Pages/Home'; // Path to your Home component
import Routes from './app/routes';
import { AuthProvider } from './context/authContext';
export default function App() {
  return (
    <View style={styles.container}>
      <AuthProvider>
        <Routes />
      </AuthProvider>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
  },
});
```

โค้ดตรงส่วนนี้จะกำหนดการแสดงผลเริ่มต้นของหน้าเลย โดยเริ่มจาก AuthProvider เพื่อให้สามารถนำฟังก์ชันและข้อมูลต่างๆของ AuthContext ไปใช้ได้ในแต่ละหน้าภายใน Routes จึงต้องมีการกำหนดตั้งแต่หน้านี้

2. Part Authentication

2.1 ส่วนของการ Sign Up

**** โค้ดนี้อยู่ใน ./context/authContext.js ****

```
const signUpWithEmail = async (email, password, roomNums, displayName, role = 'user') => {
  try {
    // สมัครผู้ใช้ใหม่
    const userCredential = await createUserWithEmailAndPassword(auth, email, password);
    console.log("Sign Up SoomBoonna");
    const user = userCredential.user;

    // เตรียมข้อมูลที่จะเก็บใน Firestore
    const userData = {
      uid: user.uid,
      displayName: displayName,
      email: user.email,
      roomNums: roomNums,
      role: role // กำหนด role (ค่าเริ่มต้นเป็น 'user')
    };
    console.log("เก็บข้อมูลแล้วนะ");
    console.log("UserId", user.uid, displayName, roomNums, user.email, role);
    // เก็บข้อมูลใน Firestore โดยใช้ UID เป็น document ID
    await setDoc(doc(db, 'users', user.uid), userData);

    console.log("User registered with display name:", displayName, "and role:", role);
  } catch (error) {
    console.error("Error during email registration:", error);
  }
};
```

ในการสมัครสมาชิกจะระบบข้อมูลจากฟิลต์ในหน้า Sign Up แล้วมาใช้ระบบของ Firebase ที่มีอยู่แล้วภายใน แต่โดยปกติระบบ Authentication ของ Firebase จะไม่สามารถดึงข้อมูลออกมาใช้ได้ง่ายๆ ตามหลักความปลอดภัยดังนั้นเราจึงต้อง นำข้อมูลที่จะเข้า ฟังก์ชันสมัครสมาชิก มาเก็บไว้ในตัวแปรและเอาไปเก็บใน Firestore ด้วยเพื่อที่จะเอา ข้อมูลเหล่า users ไปใช้ต่อได้ในหน้าถัดๆไป

**** โค้ดนี้อยู่ใน ./Pages/Register.js ****

ส่วนของ Handle Signup ที่จะรับค่าต่างๆจาก User มาเพื่อตรวจสอบก่อนที่จะเข้าสู่ฟังก์ชันสมัครสมาชิกของ Firebase

**** โค้ดนี้อยู่ใน ./Pages/Register.js ****

```
const fetchSignupStatus = async () => {
  const statusDocRef = doc(db, 'status', 'statusregister');
  const statusDoc = await getDoc(statusDocRef);

  if (statusDoc.exists()) {
    setIsSignupEnabled(statusDoc.data().status);
    console.log('Register is Open ?', statusDoc.data().status);
  } else {
    console.log('No such document!');
  }
};
```

อีกส่วนที่สำคัญที่ใช้ควบคุมหน้าการสมัครสมาชิกให้สามารถสมัครได้ หรือไม่ได้ด้วยการดึง Status จากฐานข้อมูลมาเป็น True หรือ False แล้วจึงค่อยนำไปใช้ในส่วน Handle Sign-up

2.2 ส่วนของการ Login

โค้ดนี้อยู่ใน ./app/Pages/login.js

```
const handleLogin = async () => {
  if (!email || !password) {
    setAlertMessage("กรุณากรอกอีเมลและรหัสผ่าน");
    setAlertVisible(true);
    return; // หยุดการทำงานถ้าอีเมลหรือรหัสผ่านว่าง
  }

  try {
    await auth.signInWithEmail(email, password);
    navigation.navigate('Home'); // นำทางไปหน้า Home ถ้าสำเร็จ
  } catch (error) {
    setAlertMessage(error.message);
    setAlertVisible(true);
  }
};
```

ส่วน handle login จะเป็นส่วยคอยกรองค่าจะรับมา user ว่าต้องถูกต้องตามหลัก

3. ส่วนของการ Routes

**** โค้ดนี้อยู่ใน ./app/Pages/Routes/index.js ****

```
import ReportAdd from '../Pages/Reports/ReportAdd';
import ReportEdit from '../Pages/Reports/ReportEdit';
import ReportAdminList from '../Pages/Reports/ReportAdminList';

const Stack = createNativeStackNavigator();

const Routes = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Getstart" component={Getstart} options={{ headerShown: false }}/>
        <Stack.Screen name="Login" component={Login} options={{ headerShown: false }}/>
        <Stack.Screen name="SignUp" component={Register} options={{ headerShown: false }}/>

        { /* Use BottomNavigator instead of Home to show tabs in Home */ }
        <Stack.Screen name="Home" component={BottomNavigator} options={{ headerShown: false }}/>

        <Stack.Screen name="Bill" component={Bill} options={{
          headerTitle: 'RoomRover',
          headerBackTitle: 'กลับ',
          headerStyle: {
            backgroundColor: '#29B6F6'
          },
          headerTintColor: '#fff',
        }}/>
        <Stack.Screen name="Billinfo" component={Billinfo} options={{
          headerTitle: 'RoomRover',
          headerBackTitle: 'กลับ',
          headerStyle: {
            backgroundColor: '#29B6F6'
          },
          headerTintColor: '#fff',
        }}/>
        <Stack.Screen name="News" component={News} options={{
          headerTitle: 'RoomRover',
          headerBackTitle: 'กลับ',
          headerStyle: {
```

โค้ดชุดนี้ช่วยในการจัดการนำทาง ของแอปให้สามารถไปยังหน้าต่างๆได้อย่างถูกต้อง

4. Part Fetch Data


```

useEffect(() => {
  const fetchReportData = async () => {
    try {
      const reportDocRef = doc(db, `users/${uid}/reports/${reportId}`);
      const reportDoc = await getDoc(reportDocRef);

      if (reportDoc.exists()) {
        const reportData = reportDoc.data();
        setRoom(reportData.roomNums || 'ไม่พบหมายเลขห้อง'); // ตั้งค่า room
        setReports(reportData.issue || 'ไม่พบข้อมูล'); // ตั้งค่า reports
        setStatus(reportData.status); // ตั้งค่าสถานะ
        setComment(reportData.comment || ''); // Set initial comment from the report data
      } else {
        console.log('No such report document!');
      }
    } catch (error) {
      console.error('Error fetching report data:', error);
    }
  };

  fetchReportData();
}, [reportId, uid]); // เพิ่ม uid ใน dependencies

```

ส่วนใหญ่ Code ของ Part นี้จะอยู่ในเกือบทุกหน้า แค่ดึงข้อมูลคนละตัวกัน โดยจะ fetch ข้อมูลจาก firestore โดยใช้ getDoc มาเก็บในตัวแปรและสามารถแสดงผลข้อมูลได้ในแต่ละหน้าอย่างถูกต้อง

5. Part Insert Data

```

const reportRef = collection(db, 'users', currentUser.uid, 'reports');
await addDoc(reportRef, {
  userId: currentUser.uid,
  issue: issue,
  status: 'ยังไม่แก้ไข', // Default status
  comment: '', // Default comment
  createdAt: serverTimestamp(), // Timestamp for creation
});

```

Code จุดนี้ก็จะอยู่ในบางหน้า รับหน้าทำในการส่งตัวแปรไปเก็บไว้ในฐานข้อมูลโดยใช้ฟังก์ชัน addDoc พร้อมตำแหน่งที่จะเก็บและชุดข้อมูล

การใช้งานแอปพลิเคชัน

แยกการทำงาน 2 ฝั่ง ได้แก่ admin, user

- สแกน QR Code
- ทำการ Login เข้าสู่แอปพลิเคชัน
- กรอกข้อมูล Email, Password

หากต้องการเข้าทางฝั่ง **Admin** ให้กรอก Email และ Password ตามนี้

Email: admin@gmail.com

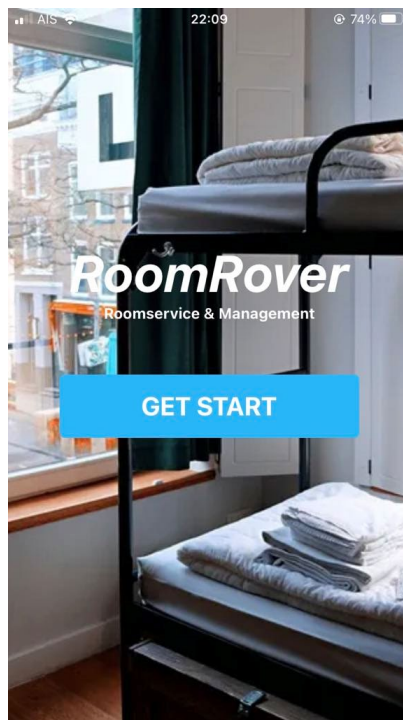
Password: 123456

หากต้องการเข้าทางฝั่ง **User** ให้กรอก Email และ Password ตามนี้

Email : user@gmail.com

Password : 123456

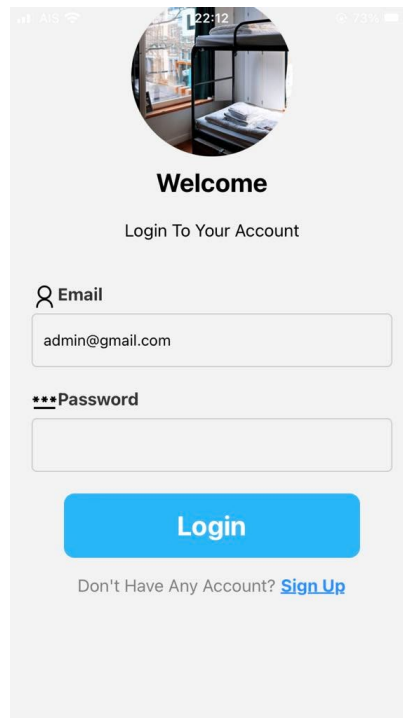
หน้าตาของฟังก์ชันและการทำงานของแอปพลิเคชันในแต่ละ Role



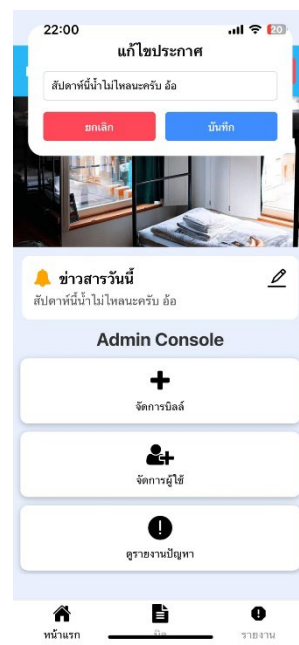
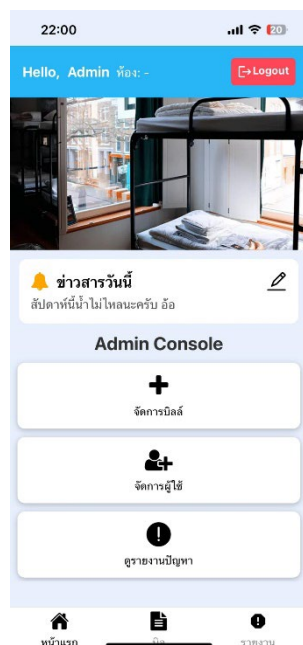
**** หน้าเริ่มต้นของ application ****

Admin

1. สามารถ Login เข้าสู่ระบบได้โดยผ่านหน้า Login



2. หน้าหลักของแอปพลิเคชัน โดยจะแสดงรายละเอียดดังนี้
 - 2.1 แสดงชื่อของ admin
 - 2.2 แสดง console ต่างๆที่ admin สามารถจัดการได้ เช่น ข่าวสาร การจัดการบิล การจัดการผู้ใช้ และการดูรายการปัญหา ดังภาพต่อไปนี้



หน้า console ของ admin ที่เป็นหน้าหลัก โดยสามารถแก้ไขข่าวสารประจำวันให้แก่ User ได้



หน้าของ admin ที่สามารถตรวจสอบจำนวนและตรวจสอบข้อมูลบิลของ User ทั้งหมดได้

22:08 20

< กลับ RoomRover

เพิ่มข้อมูลบิล

ชื่อผู้เช่า
Chin

หมายเลขห้องพัก
101

ค่าห้องพัก
3000

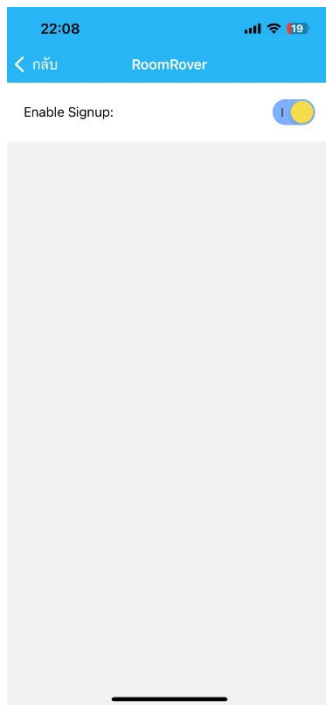
ค่าไฟฟ้า (หน่วยที่ใช้)
จำนวนหน่วยไฟฟ้าที่ใช้

ค่าน้ำ
ค่าน้ำ

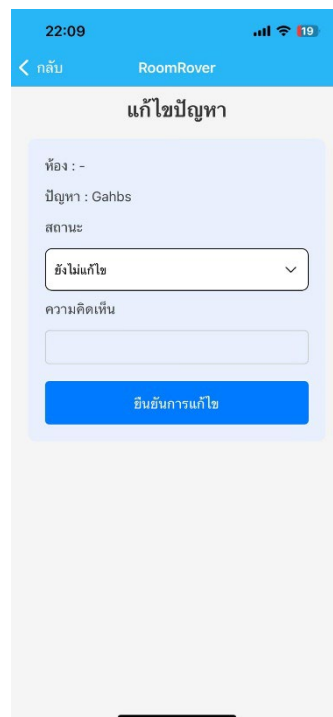
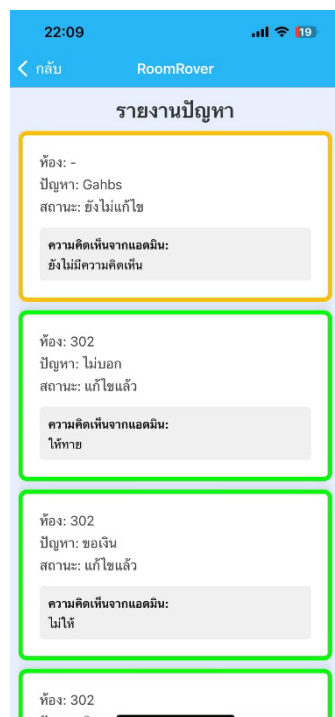
เดือน
เลือกเดือน

ยืนยันการเพิ่มข้อมูลบิล

หน้าของ admin ที่สามารถเพิ่มบิลที่ต้องให้ user ชำระในแต่ละเดือนได้



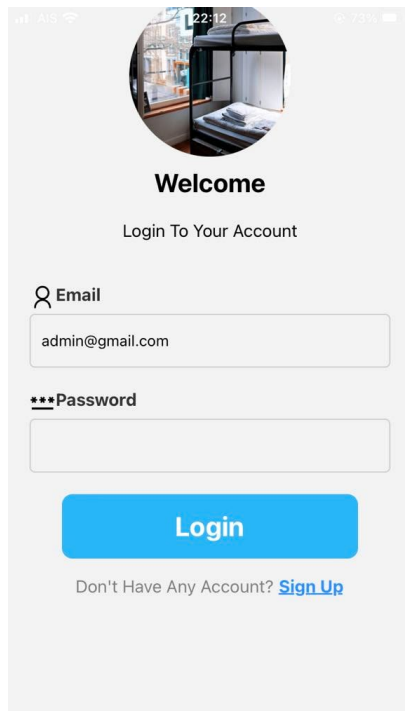
หน้าจอของ admin ที่สามารถอนุมัติบิลให้ user สมัครสมาชิกได้



หน้าจอของ admin ที่สามารถอ่านปัญหาที่ user รายงานได้และแก้ไขให้กับ user ได้

User

1. สามารถ Login เข้าสู่ระบบได้โดยผ่านหน้า Login และสามารถสมัครสมาชิกได้หาก Admin ได้อนุมัติแล้ว



22:12 73%

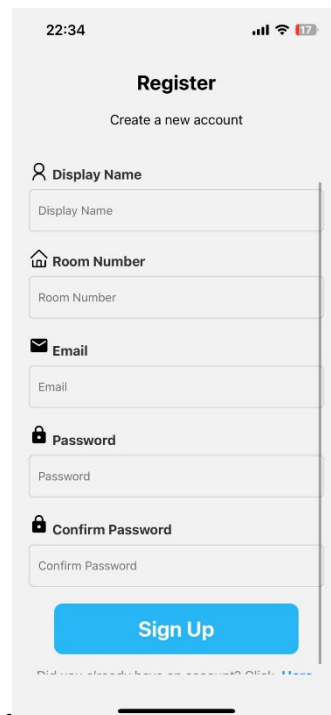
Welcome

Login To Your Account

*****Password**

Login

Don't Have Any Account? [Sign Up](#)



22:34 17%

Register

Create a new account

Sign Up

Please check the account is correct? Click [Back](#)

2. หน้าหลักของแอปพลิเคชัน จะแสดงรายละเอียดดังนี้
 - 2.1 แสดงชื่อของ User และบอกหมายเลขห้อง
 - 2.2 แสดงหน้าหลัก User โดยสามารถดูข้อมูลต่างๆได้ เช่น ข่าวสาร บิล กราฟแสดงค่าไฟ ย้อนหลัง 6 เดือนได้ และ เขียนรายการปัญหาได้ ดังภาพต่อไปนี้



หน้าของ user ที่สามารถดูข้อมูลกราฟแสดงค่าไฟย้อนหลัง 6 เดือนได้

22:31 RoomRover

ชื่อผู้เช่า: User

บิลค่าเช่า:	เดือนไม่ทราบ	ชำระแล้ว
บิลค่าเช่า:	เดือนกรกฎาคม	ชำระแล้ว
บิลค่าเช่า:	เดือนสิงหาคม	ชำระแล้ว
บิลค่าเช่า:	เดือนกันยายน	ยังไม่ชำระ
บิลค่าเช่า:	เดือนตุลาคม	ยังไม่ชำระ

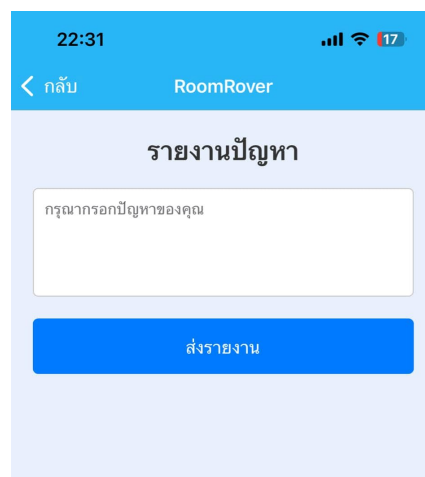
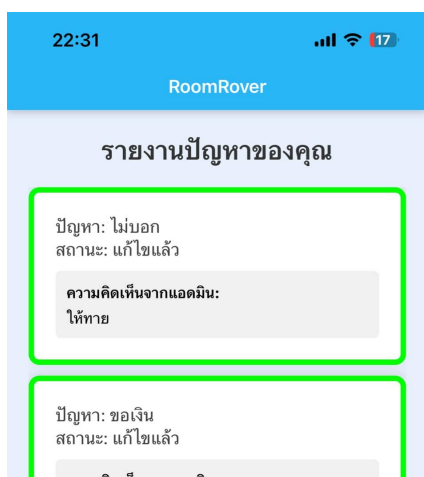
22:31 RoomRover

กลับ

บิลค่าเช่า

สถานะ	ยังไม่ชำระ
รายการ	
ค่าเช่าห้อง	3000 บาท
ค่าน้ำ	200 บาท
ค่าไฟฟ้า	1040 บาท
รวมทั้งสิ้น	4240 บาท
ชำระเงิน	

หน้าของ user ที่สามารถเช็คบิลที่จ่ายไปแล้วหรือค้างชำระโดยหากค้างชำระสามารถชำระเงินได้



หน้าของ user ที่สามารถดูปัญหาทั้งหมดได้และยังสามารถเพิ่มการรายงานปัญหาได้