[Department of Electrical and Computer Engineering]
North South University

Senior Design Project

[BDRIDER]

[A Bicycle Renting App]

Team Members

Name	Id
Md. AzizulHaqueSrejan	1511031642
JayantiSaha	1421176043
A.F. MimOnika	1410553042
Hridi Ahmed	1231314042

[Faculty Advisor]

Dr.Shazzad Hossain

Associate Professor

Department of Electrical & Computer Engineering

North South UniversityDhaka,Bangladesh

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' names & Signatures

1. Md Azizul Haque Srejan

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _

2. Jayanti Saha

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _

3. A.F. Mim Onika

_ _ _ _ _ _ _ _ _ _ _ _ _ _

4. Hridi Ahmed

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _

APPROVAL

We, Md. Azizul Haque Srejan (1511031642), Jayanti Saha (1421176043), A.F. Mim Onika (141055304), Hridi Ahmed(1231314042), members 0f CSE/EEE-499(Senior Design) from Section -5 from Electrical and Computer Engineering department of North South University has done our project named BDRIDER (A Bicycle Renting App) under the supervision of Dr. Shazzad Hossain, Associate Professor of ECE department. We have fulfilled all terms and condition of this course and completed our project.

Supervisor's Signature

- -

Dr. Shazzad Hossain
Associate Professor

Department of Electrical & Computer Engineering

North South University

Dhaka,Bangladesh

Chairman's Signature

- - - - - - - - - - - - - - - - - - - -

Dr. K.M.A. Salam
Professor & Chairman

Department of Electrical & Computer Engineering

North South University

Dhaka ,Bangladesh

ACKNOWLEDGEMENT

By mercy of the Almighty we have completed our senior design capstone project entitled “BDRIDER (A Bicycle Renting App)”. First of all we want to thank our advisor, Dr. Shazzad Hossain for his continuous support in our capstone project throughout the whole 499A and 499B with his patience, motivation, and immense knowledge. His guidance helped us in through the time to finishing our project. We would also like to thank North South University, Dhaka, Bangladesh for providing an opportunity in our curriculum which helped us to learn so much during proceeding with our project. And finally, we would like to thank our friends and family members for their inspiration and guidance which kept us focused and motivated.

ABSTRACT

Our aim was to build a system which will reduce human traveling cost and is also environment friendly. And we wanted to make traveling easy to our people. Considering some problems like traffic jam, lack of public transport, high traveling cost of private transport and environment pollution of our country and specially of the Dhaka city, we have built a system for traveling which can be used easily by an android app, that can reduce people's traveling cost and also do not have any bad impact to our environment. In our project we are introducing BDRIDER.apk an android app based software which can be used to rent bicycle. We have developed an android app where there will be two type of users- users and owners. A user can rent a bicycle from our garages which using our and owners can share their bicycle to our system.

Table of Contents

Chapter1
Overview

1.1 Introduction

Bangladesh is a developing country with lots of problem due to its huge population density. Environment pollution, traffic jam, low income etc are the common problems here. So, we have tried to solve some of the problems through our project, by trying to reduce human travelling cost in an environmental friendly way making travelling easier to people. We have build an android based system from where people can rent and lend bicycle as their convenience. Now a days app has become the easiest and most popular way in terms of availing any service. In our system bicycle owner has the opportunity to share their bicycle to our service and earn from there. Here we have a lots of garage at different location which a user can find in maps using our application. They can rent a bicycle from a garage and can checkout to other garage. Those owner who have shared their bicycle to our system will get account balance instantly and can check their account by using our app.

1.2 Project Definition

In our project we are introducing a system from where people can rent bicycle by using an android app. First of all a user has to create their account by giving their particular information like- name, address, mobile number, nid number etc. After creating a account to our system a user can use our service. When a user will login to their account they can see thee Google map and nearest bicycle garage. Then they can choice their desire cycle from any garage and can start their ride by scanning the qr code of that garage and bicycle. Then they will get a unlock code in their app by which they can unlock their chosen bicycle. After completing their ride they can check out the bicycle to other nearest garage or to the same garage. Then they have to scan that garage code and can checkout. After checking out the user will get their total time and total bill and can pay bill through cash or bkash. A owner can check their account balance and cycle information by using the app. A supervisors can monitor any particular garage by our app.

1.3 Motivation/Purpose of this project

Environment pollution, traffic jam, low income etc are common problems of our country. There was a time when commuters suffered traffic congestion only on the main city streets, but now it starts right from one's doorstep. Traffic jam has turned daily trips into nightmares. According to a World Bank report, in the last 10 years, the average traffic speed in Dhaka has dropped from 21 kilometres per hour (kmph) to 7 kmph, and by 2035, the speed might drop to 4kmph, which is slower than the walking speed. Another study, commissioned by Brac Institute of Government and Development, says traffic congestion in Dhaka eats up around 5 million working hours every day and costs the country USD 11.4 billion every year. The financial loss is a calculation of the cost of time lost in traffic congestion and the money spent on operating vehicles for the extra

hours. Again Bangladesh has experienced a sharp increase in air pollution since 2010 as 81 percent population are exposed to household air pollution from solid fuel burning, says a new global study report.

“Pakistan, Bangladesh and India, on the other hand, have experienced the steepest increases in air pollution levels since 2010,” adds the report styled Global Air/2018 Report published on Tuesday by the Health Effects Institute (HEI).More than 95 percent of global population are breathing unhealthy air and the poorest nations are the worst victims, according to the report, which further says long-term exposure to air pollution contributed to an estimated 6.1 million deaths around the globe in 2016. With increasing improvements in the transportation system of our country, and the constant developments in our infrastructure, we may not be far away from the day when being stuck in the traffic for hours will not be an everyday thing.

Starting from rickshaws and buses to the newly introduced Ubers, we have seen a lot of changes till date. It was Public Transport day on November 10. So, we went around the city to see how far we have actually come, in terms of growth in the public transportation.

Rickshaws

For us Dhaka dwellers, rickshaw has to be the best mode of transportation. Roaming around university campuses and narrow alleys of the city on rickshaw, has always been a favourite pastime for many of us.

The availability of rickshaws and cheap rates have always appealed to us, Dhakaites. And with the introduction of the yellow rickshaws around the diplomatic zone, after the terror attack in July, students and commuters of all ages find it easier to travel from one place to another. On August 10, 500 new yellow rickshaws were commissioned for around Niketon, Gulshan, Banani, and Baridhara area. Working around the clock in three shifts, 1,500 rickshaw pullers have been recruited and trained to act as police informants. These rickshaw pullers have unique coloured uniforms based on their area of operation with logos of the respective housing societies. The fares of twenty destinations have been fixed and they are displayed on the number plates. The rates range from Tk15 to Tk50. Moreover, these plates are retro reflective which can be easily detected by the CC cameras. Rickshaw pullers in these areas have been assigned a hotline number to report any suspicious activities. Hence travelling on rickshaws has become a lot safer.

The number of rickshaws launched however, is inadequate. Although 1000 of these vehicles were added to the existing 500, it still seems insufficient as at least 5000 more are needed. We also cannot overlook the fact that they were never allowed to drive on the main roads. Yellow rickshaws are limited to certain areas, it gives the ordinary rickshaw pullers a chance to demand very high prices. The scarcity of rickshaws becomes a major crisis on rainy days and particularly during the rush hours.

Buses and CNG-driven auto rickshaws

If you live in Dhaka, you cannot help but notice people running after buses, or bargaining with autorickshaw drivers over the fare. Reckless bus drivers making their way through heavy traffic in the morning, that too on the wrong side, is also a common scenario. Just as rickshaws are widely used for short distances, buses and auto rickshaws (known as CNGs), are mainly used for longer routes.

Travelling by the bus is convenient among young students and commuters, due mainly to the affordable fares and accessibility. Some of these buses even offer special discounts for university and college going students. But one cannot deny that extensive routes and numerous stoppages have always been a problem when travelling by the bus. Twenty new air-conditioned buses, run by Private Operators have been launched recently. Though the fixed routes from Mohakhali to Gulshan through Badda link road, have been a difficulty for most of the passengers. And as most of the buses carry way more passengers than they're supposed to, problems of overloading will always arise, unless the number of buses are increased.

Overtime, CNG-driven auto rickshaws have gained popularity for a certain level of safety. Although there have been mishaps every now and then. Measures have been taken to make CNGs more safe and cost effective. Since the fares have been assigned according to the distance, people do not have to face the hassle of bargaining. However it still takes a lot of convincing, as these drivers tend to ask for at least Tk10-Tk20 more than the actual fare.

1.4 Project Goal

Our goal is to reduce human suffering and environmental pollution by making a traveling system which is easy to use. We have built a system for traveling which can be used easily by an android app .Which will reduce peoples traveling cost and also do not create any bad impact over our environment. Now a day's app has become the easiest way to use any service and become very popular. So, in our project we are introducing BDRIDER.apk an android app software which can be used to rent bicycle. Bicycling is also good for our health. Because it will help users to take some physical exercise during travelling. In our system bicycle owner has the opportunity to share their bicycle to our service to income extra money.

1.5 Summary

Our system will be useful to the users because they can rent a bicycle at any time from our garage and can travel to other place comfortably. They can checkout bicycle to other garage near to their destination. Bicycle owner has the opportunity to share their bicycle to our service to earn extra money. Supervisor can monitor his particular garage by using this apps. So, our system has brought solution to many problems under one simple application.

Chapter 2
Features

2.1 Introduction

In this section we are going to talk about the features of our application. Our application provides a range of features for the app users who want to rent or lend bicycle. We are going to discuss about them in this section.

2.2 Feature list

- Signing up as user
- Signing up as owner
- Login as user
- Login as owner
- Login as supervisor
- Choosing a garage
- Rent bicycle
- Choose garage
- Choose bicycle
- Scan QR code for garage
- Scan QR code for bicycle
- Unlock lock
- Time and rent calculation
- Check balance
- Checkout
- Rating the application

2.3 Feature Description

2.3.1 Signing up as Owner and User

After downloading and installing the application the users who want to rent or lend bicycle needs to sign up. This is done in two different ways- one as user and one as owner. Users are the people who want to rent bicycle through our application and owners are the people who already own bicycles and want to lend them.

While signing up as user the users will have to provide their username, phone number and email address. A verification code is sent to that number which has to be inputted in the application for the sign up to finish.

While signing up as owner the owner has to input his/her name and the code provided to them while registering the bicycle. After that the owner has to input his name, phone number, cycle model and cycle number. Sign up is done once they have inputted all the necessary credentials.

2.3.2 Signing is as User, Owner and Supervisor

Once the user has signed up he/she has to sign in into the the app. For signing in as user, owner or supervisor they have to input their username and phone number.

2.3.3 Choosing a Garage

After signing in as a user, the user can see a map where they can see the garages near them. Selecting the most convenient garage the address of the garage would be shown there. Users can navigate themselves to their desirable garage from the address shown there.

2.3.4 Renting Bicycle

After reaching the garage the user can rent bicycle for which they have to scan the garage QR code and also the QR code for the selected bicycle. After both the scans are done a pass code appears at the app which has to be imputed to unlock the lock. Once the lock is unlocked the cycle is ready to be rented.

2.3.5 Time and Rent Calculation

Once the bicycle is rented the timer starts. While checking out the timer stops. And the amount to be payed is shown to the user. The amount payable is calculated by multiplying the number of minutes elapsed by BDT0.50.

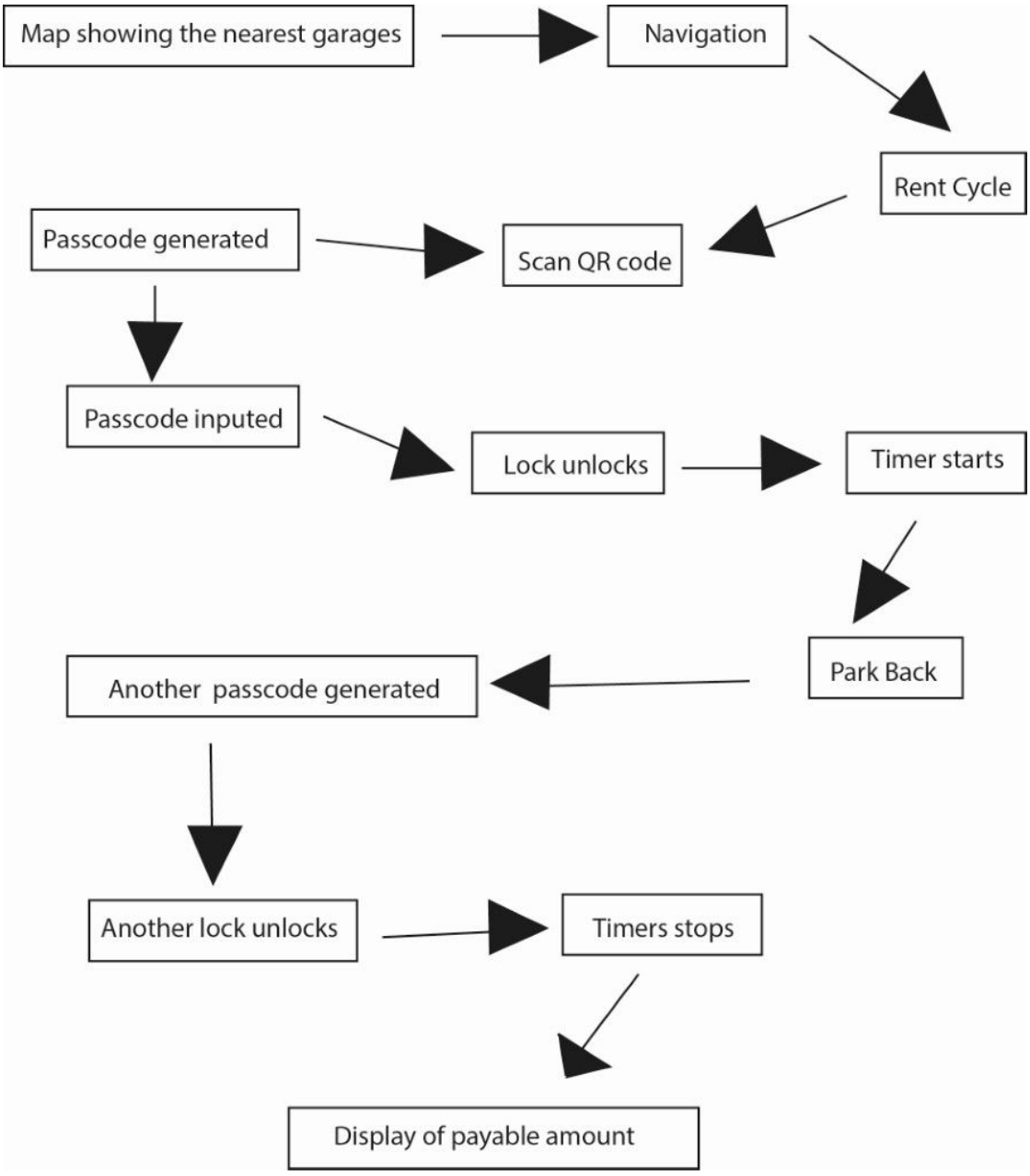
2.3.6 Checking Bill

Owners who have rented their bicycle can check the number of times their bicycle has been rented and the amount of money that has been added to their account.

2.3.7 Rating the Application

Our application also has the feature of rating the app according to users likeability towards the app. Users can rate the app from one star, carrying the least likeness at the likert scale, to five star, carrying the maximum likeness at the likert scale.

2.3.8 Flowchart



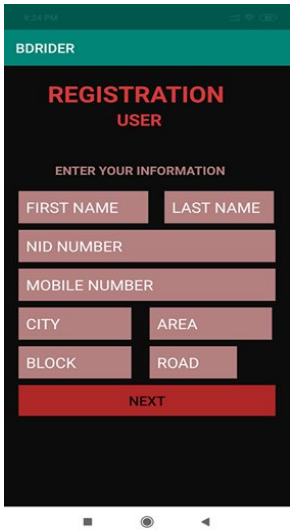
2.3.9 Screenshots

User

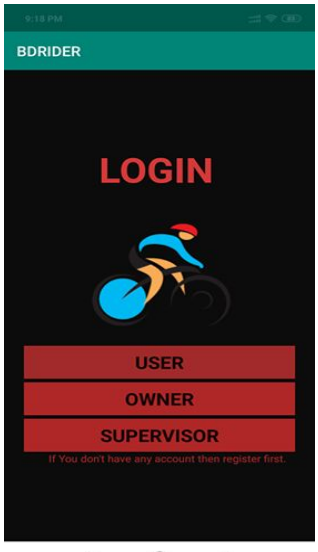
1

2

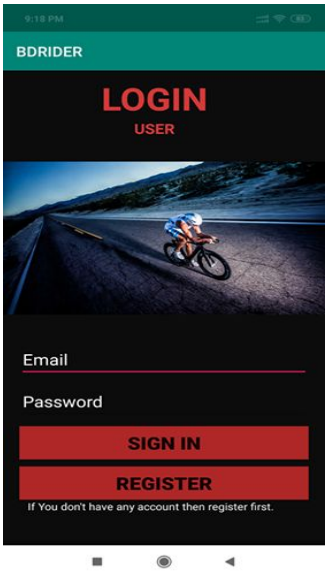
3



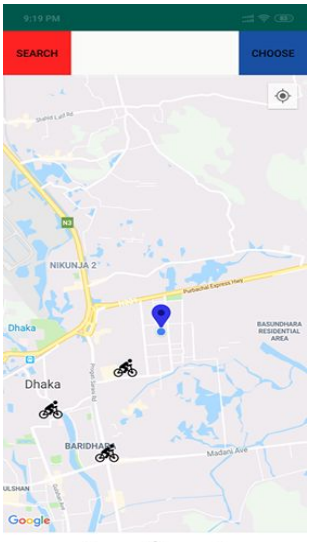
4



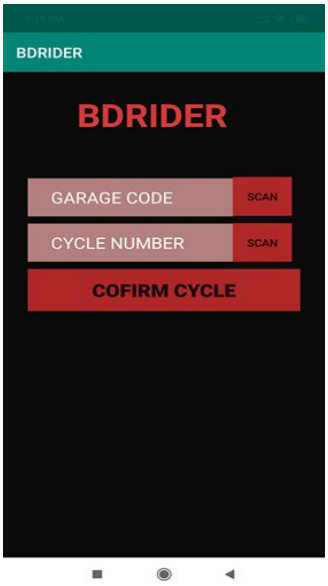
5



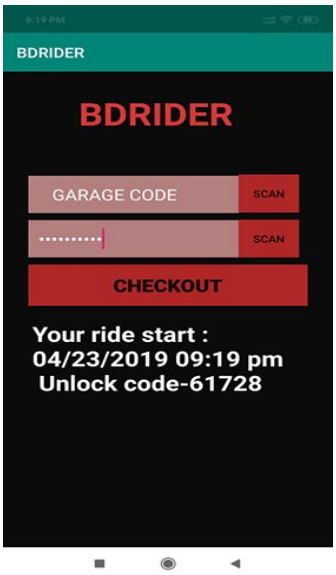
6



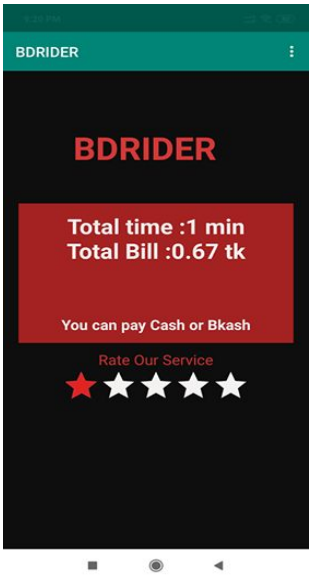
7



8

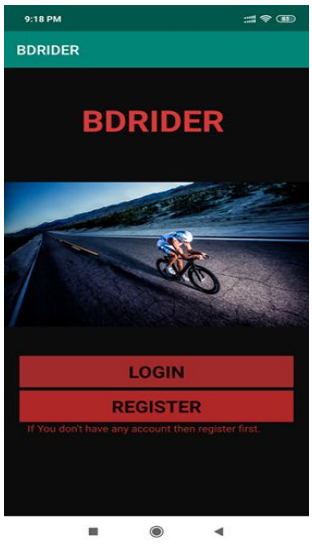


9

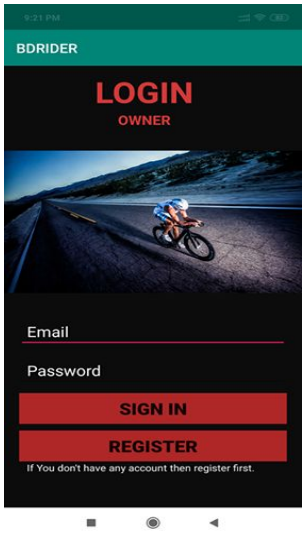


Owner and Supervisor

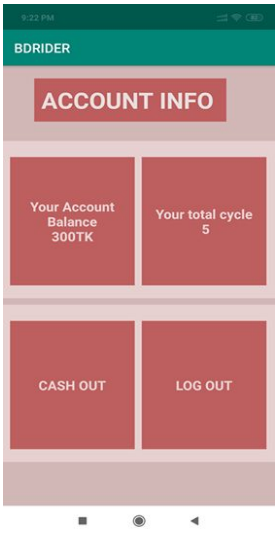
1



2



3



Chapter 3
Software Implementation

3.1 Introduction

In this section we will discuss about the implementation of our system. Our system is basically based on application. Users can use our whole system by using only an app. There are lots of features for users, bicycle owners and garage supervisor. In this section we will talk about how we implemented these features.

3.2 Feature Implementation

In our application there are lot of features for users. Like

a) Create account,
b) Google maps,
c) Scan code,
d) To get unlock key etc.

3.2.1 Create Account

To use our service users have to create an account to our system. They have to provide necessary information about them. After creating an account any user can use our system by login to their account. We have used Firebase authentication and Firebase database for this create account and login system.

3.2.2 Firebase Authentication

Authenticate with Firebase on Android using a Phone Number

You can use Firebase Authentication to sign in a user by sending an SMS message to the user's phone. The user signs in using a one-time code contained in the SMS message.

The easiest way to add phone number sign-in to your app is to use Firebase UI, which includes a drop-in sign-in widget that implements sign-in flows for phone number sign-in, as well as password-based and federated sign-in. This document describes how to implement a phone number sign-in flow using the Firebase SDK.

Phone numbers that end users provide for authentication will be sent and stored by Google to improve our spam and abuse prevention across Google services, including but not limited to Firebase. Developers should ensure they have appropriate end-user consent prior to using the Firebase Authentication phone number sign-in service.

Before you begin

If you haven't already, add Firebase to your Android project.

Add the dependency for the Firebase Authentication Android library to your module (app-level) Gradle file (usually app/build.gradle):

Implementation 'com.google.firebase:firebase-auth:16.2.1'

If you haven't yet connected your app to your Firebase project, do so from the Firebase console.

If you haven't already set your app's SHA-1 hash in the Firebase console, do so. See Authenticating Your Client for information about finding your app's SHA-1 hash.

Also, note that phone number sign-in requires a physical device and won't work on an emulator.

Security concerns

Authentication using only a phone number, while convenient, is less secure than the other available methods, because possession of a phone number can be easily transferred between users. Also, on devices with multiple user profiles, any user that can receive SMS messages can sign in to an account using the device's phone number.

If you use phone number based sign-in in your app, you should offer it alongside more secure sign-in methods, and inform users of the security tradeoffs of using phone number sign-in.

Enable Phone Number sign-in for your Firebase project

To sign in users by SMS, you must first enable the Phone Number sign-in method for your **Firebase project:**

In the Firebase console, open the Authentication section.

On the Sign-in Method page, enable the Phone Number sign-in method.

Firebase's phone number sign-in request quota is high enough that most apps won't be affected. However, if you need to sign in a very high volume of users with phone authentication, you might need to upgrade your pricing plan. See the pricing page.

Send a verification code to the user's phone

To initiate phone number sign-in, present the user an interface that prompts them to type their phone number. Legal requirements vary, but as a best practice and to set expectations for your users, you should inform them that if they use phone sign-in, they might receive an SMS message for verification and standard rates apply.

Then, pass their phone number to the PhoneAuthProvider.verifyPhoneNumber method to request that Firebase verify the user's phone number. For example:

```
PhoneAuthProvider.getInstance().verifyPhoneNumber(
        phoneNumber,        // Phone number to verify
        60,                 // Timeout duration
        TimeUnit.SECONDS,   // Unit of timeout
        this,               // Activity (for callback binding)
        mCallbacks);        // OnVerificationStateChangedCallbacks
```

The verifyPhoneNumber method is reentrant: if you call it multiple times, such as in an activity's onStartmethod, the verifyPhoneNumber method will not send a second SMS unless the original request has timed out.
You can use this behavior to resume the phone number sign in process if your app closes before the user can sign in (for example, while the user is using their SMS app). After you call verifyPhoneNumber, set a flag that indicates verification is in progress. Then, save the flag in your Activity's onSaveInstanceState method and restore the flag in onRestoreInstanceState. Finally, in your Activity's onStart method, check if verification is already in progress, and if so, call verifyPhoneNumber again. Be sure to clear the flag when verification completes or fails (see Verification callbacks).
To easily handle screen rotation and other instances of Activity restarts, pass your Activity to the verifyPhoneNumber method. The callbacks will be auto-detached when the Activity stops, so you can freely write UI transition code in the callback methods.
The SMS message sent by Firebase can also be localized by specifying the auth language via the setLanguageCode method on your Auth instance.

```
auth.setLanguageCode("fr");
// To apply the default app language instead of explicitly setting it.
// auth.useAppLanguage();
```

MainActivity.java

When you call PhoneAuthProvider.verifyPhoneNumber, you must also provide an instance of OnVerificationStateChangedCallbacks, which contains implementations of the callback functions that handle the results of the request. For example:

```
mCallbacks = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

    @Override
    public void onVerificationCompleted(PhoneAuthCredential credential) {
        // This callback will be invoked in two situations:
        // 1 - Instant verification. In some cases the phone number can be instantly
        //     verified without needing to send or enter a verification code.
        // 2 - Auto-retrieval. On some devices Google Play services can automatically
        //     detect the incoming verification SMS and perform verification without
        //     user action.
        Log.d(TAG, "onVerificationCompleted:"  + credential);

        signInWithPhoneAuthCredential(credential);
```

```
    }

    @Override
    public void onVerificationFailed(FirebaseException e) {
        // This callback is invoked in an invalid request for verification is made,
        // for instance if the the phone number format is not valid.
        Log.w(TAG, "onVerificationFailed", e);

        if (e instanceof FirebaseAuthInvalidCredentialsException) {
            // Invalid request
            // ...
        } else if (e instanceof FirebaseTooManyRequestsException) {
            // The SMS quota for the project has been exceeded
            // ...
        }

        // Show a message and update the UI
        // ...
    }

    @Override
    public void onCodeSent(String verificationId,
                           PhoneAuthProvider.ForceResendingToken token) {
        // The SMS verification code has been sent to the provided phone number, we
        // now need to ask the user to enter the code and then construct a credential
        // by combining the code with a verification ID.
        Log.d(TAG, "onCodeSent:" + verificationId);

        // Save verification ID and resending token so we can use them later
        mVerificationId = verificationId;
        mResendToken = token;

        // ... }};
```

Verification callbacks

In most apps, you implement the onVerificationCompleted, onVerificationFailed, and onCodeSentcallbacks. You might also implement onCodeAutoRetrievalTimeOut, depending on your app's requirements.

onVerificationCompleted(PhoneAuthCredential)

This method is called in two situations:
Instant verification: in some cases the phone number can be instantly verified without needing to send or enter a verification code.
Auto-retrieval: on some devices, Google Play services can automatically detect the incoming verification SMS and perform verification without user action. (This capability might be unavailable with some carriers.)
In either case, the user's phone number has been verified successfully, and you can use the PhoneAuthCredential object that's passed to the callback to sign in the user.
onVerificationFailed(FirebaseException)
This method is called in response to an invalid verification request, such as a request that specifies an invalid phone number or verification code.
onCodeSent(String verificationId, PhoneAuthProvider.ForceResendingToken)
Optional. This method is called after the verification code has been sent by SMS to the provided phone number.
When this method is called, most apps display a UI that prompts the user to type the verification code from the SMS message. (At the same time, auto-verification might be proceeding in the background.) Then, after the user types the verification code, you can use the verification code and the verification ID that was passed to the method to create a PhoneAuthCredential object, which you can in turn use to sign in the user. However, some apps might wait until onCodeAutoRetrievalTimeOut is called before displaying the verification code UI (not recommended).
onCodeAutoRetrievalTimeOut(String verificationId)
Optional. This method is called after the timeout duration specified to verifyPhoneNumber has passed withoutonVerificationCompleted triggering first. On devices without SIM cards, this method is called immediately because SMS auto-retrieval isn't possible.
Some apps block user input until the auto-verification period has timed out, and only then display a UI that prompts the user to type the verification code from the SMS message (not recommended).
Create a PhoneAuthCredential object
After the user enters the verification code that Firebase sent to the user's phone, create a PhoneAuthCredentialobject, using the verification code and the verification ID that was passed to the onCodeSent oronCodeAutoRetrievalTimeOut callback. (When onVerificationCompleted is called, you get aPhoneAuthCredential object directly, so you can skip this step.)
To create the PhoneAuthCredential object, call PhoneAuthProvider.getCredential:
PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationId, code);
PhoneAuthActivity.java
To prevent abuse, Firebase enforces a limit on the number of SMS messages that can be sent to a single phone number within a period of time. If you exceed this limit, phone number verification

requests might be throttled. If you encounter this issue during development, use a different phone number for testing, or try the request again later.

Sign in the user

After you get a PhoneAuthCredential object, whether in the onVerificationCompleted callback or by callingPhoneAuthProvider.getCredential, complete the sign-in flow by passing the PhoneAuthCredential object to FirebaseAuth.signInWithCredential:

```
private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
   mAuth.signInWithCredential(credential)
         .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
               if (task.isSuccessful()) {
                  // Sign in success, update UI with the signed-in user's information
                  Log.d(TAG, "signInWithCredential:success");

                  FirebaseUser user = task.getResult().getUser();
                  // ...
               } else {
                  // Sign in failed, display a message and update the UI
                  Log.w(TAG, "signInWithCredential:failure", task.getException());
                  if (task.getException() instanceof FirebaseAuthInvalidCredentialsException) {
                     // The verification code entered was invalid
                  }
               }
            }
         });
}
```

Test with whitelisted phone numbers

You can whitelist phone numbers for development via the Firebase console. Whitelisting phone numbers provides these benefits:

Test phone number authentication without consuming your usage quota.

Test phone number authentication without sending an actual SMS message.
Run consecutive tests with the same phone number without getting throttled. This minimizes the risk of rejection during App store review process if the reviewer happens to use the same phone number for testing.
Test readily in development environments without any additional effort, such as the ability to develop in an iOS simulator or an Android emulator without Google Play Services.
Write integration tests without being blocked by security checks normally applied on real phone numbers in a production environment.
Phone numbers to whitelist must meet these requirements:
Make sure you use fictional numbers that do not already exist. Firebase Authentication does not allow you to whitelist existing phone numbers used by real users. One option is to use 555 prefixed numbers as US test phone numbers, for example: *+1 650-555-3434*
Phone numbers have to be correctly formatted for length and other constraints. They will still go through the same validation as a real user's phone number.
You can add up to 10 phone numbers for development.
Use test phone numbers/codes that are hard to guess and change those frequently.
Whitelist phone numbers and verification codes
In the Firebase console, open the Authentication section.
In the Sign in method tab, enable the Phone provider if you haven't already.
Open the Phone numbers for testing accordion menu.
Provide the phone number you want to test, for example: *+1 650-555-3434*.
Provide the 6-digit verification code for that specific number, for example: *654321*.
Add the number. If there's a need, you can delete the phone number and its code by hovering over the corresponding row and clicking the trash icon.

Manual testing
You can directly start using a whitelisted phone number in your application. This allows you to perform manual testing during development stages without running into quota issues or throttling. You can also test directly from an iOS simulator or Android emulator without Google Play Services installed.
When you provide the whitelisted phone number and send the verification code, no actual SMS is sent. Instead, you need to provide the previously configured verification code to complete the sign in.
On sign-in completion, a Firebase user is created with that phone number. The user has the same behavior and properties as a real phone number user, and can access Realtime Database/Cloud Firestore and other services the same way. The ID token minted during this process has the same signature as a real phone number user.
Because the ID token for the whitelisted phone number has the same signature as a real phone number user, it is important to store these numbers securely and to continuously recycle them.

Another option is to set a test role via custom claims on these users to differentiate them as fake users if you want to further restrict access.

Integration testing

In addition to manual testing, Firebase Authentication provides APIs to help write integration tests for phone auth testing. These APIs disable app verification by disabling the reCAPTCHA requirement in web and silent push notifications in iOS. This makes automation testing possible in these flows and easier to implement. In addition, they help provide the ability to test instant verification flows on Android.

Make sure app verification is not disabled for production apps and that no whitelisted phone numbers are hardcoded in your production app.

On Android you can readily use your whitelisted phone numbers without any additional API calls. Calling verify Phone Number with a white listed number triggers the onCodeSent callback, in which you'll need to provide the corresponding verification code. This allows testing in Android Emulators.

```
String phoneNum = "+16505554567";
String testVerificationCode = "123456";

// Whenever verification is triggered with the whitelisted number,
// provided it is not set for auto-retrieval, onCodeSent will be triggered.


PhoneAuthProvider.getInstance().verifyPhoneNumber(
      phoneNum, 30L /*timeout*/, TimeUnit.SECONDS,
      this, new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

         @Override
         public void onCodeSent(String verificationId,
                                PhoneAuthProvider.ForceResendingToken forceResendingToken) {
            // Save the verification id somewhere
            // ...

            // The corresponding whitelisted code above should be used to complete sign-in.
            MainActivity.this.enableUserManuallyInputCode();
         }

         @Override
         public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {
```

```
            // Sign in with the credential
            // ...
        }

        @Override
        public void onVerificationFailed(FirebaseException e) {
            // ...
        }

    });
```

Additionally, you can test auto-retrieval flows in Android by setting the whitelisted number and its corresponding verification code for auto-retrieval by calling setAutoRetrievedSmsCodeForPhoneNumber.
When verifyPhoneNumber is called, it triggers onVerificationCompleted with the PhoneAuthCredentialdirectly. This works only with whitelisted phone numbers.
Make sure this is disabled and no whitelisted phone numbers are hardcoded in your app when publishing your application to the Google Play store.

```
// The test phone number and code should be whitelisted in the console.
String phoneNumber = "+16505554567";
String smsCode = "123456";

FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
FirebaseAuthSettings firebaseAuthSettings = firebaseAuth.getFirebaseAuthSettings();

// Configure faking the auto-retrieval with the whitelisted numbers.
firebaseAuthSettings.setAutoRetrievedSmsCodeForPhoneNumber(phoneNumber, smsCode);

PhoneAuthProvider phoneAuthProvider = PhoneAuthProvider.getInstance();
phoneAuthProvider.verifyPhoneNumber(
        phoneNumber,


  TimeUnit.SECONDS,
        this, /* activity */
        new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
```

```
    @Override
    public void onVerificationCompleted(PhoneAuthCredential credential) {
        // Instant verification is applied and a credential is directly returned.
        // ...
    }

    // ...
});
```

Next steps

After a user signs in for the first time, a new user account is created and linked to the credentials—that is, the user name and password, phone number, or auth provider information—the user signed in with. This new account is stored as part of your Firebase project, and can be used to identify a user across every app in your project, regardless of how the user signs in.

In your apps, you can get the user's basic profile information from the FirebaseUser object. See Manage Users.

In your Firebase Realtime Database and Cloud Storage Security Rules, you can get the signed-in user's unique user ID from the auth variable, and use it to control what data a user can access.

You can allow users to sign in to your app using multiple authentication providers by linking auth provider credentials to an existing user account.

To sign out a user, call signOut:

FirebaseAuth.getInstance().signOut();

3.2.3 Firebase Realtime Database

Store and sync data with our NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline.

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

Try Cloud Firestore, the latest realtime, scalable NoSQL database from Firebase and Google Cloud Platform. Learn more about the differences between Cloud Firestore and Realtime Database.

Key capabilities

Realtime

Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.

Offline

Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.

Accessible from Client Devices

The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.

Scale across multiple databases

With Firebase Realtime Database on the Blaze pricing plan, you can support your app's data needs at scale by splitting your data across multiple database instances in the same Firebase project. Streamline authentication with Firebase Authentication on your project and authenticate users across your database instances. Control access to the data in each database with custom Firebase Realtime Database Rules for each database instance.

How does it work?

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

Implementation path

1. Integrate the Firebase Realtime Database SDKs - Quickly include clients via Gradle, CocoaPods, or a script include.
2. Create Realtime Database References - Reference your JSON data, such as "users/user: 1234 /phone_number" to set data or subscribe to data changes.
3. Set Data and Listen for Changes - Use these references to write data or subscribe to changes.
4. Enable Offline Persistence - Allow data to be written to the device's local disk so it can be available while offline.
5. Secure your data - Use Firebase Realtime Database Security Rules to secure your data.

Looking to store other types of data?

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. To learn more about the differences between database options, see Choose a database: Cloud Firestore or Realtime Database.

Firebase Remote Config stores developer specified key-value pairs to change the behavior and appearance of your app without requiring users to download an update.

Firebase Hosting hosts the HTML, CSS, and JavaScript for your website as well as other developer-provided assets like graphics, fonts, and icons.

Cloud Storage stores files such as images, videos, and audio as well as other user-generated content.

Next steps:

Set data and listen for changes using the iOS, Android, Web, Admin SDKs, or the REST API.

Add Firebase Realtime Database to your iOS, Android, or Web app.

Learn about how to secure your files using Firebase Realtime Database Rules.

3.2.4 Google Map

Here we will discuss the procedure about how we have added Google map to our Android app.

Steps to add google map on android app-

Add the Google Play services package to Android Studio.

Clone or download the Google Maps Android API v2 Samples repository if you didn't do that when you started reading this tutorial.
Import the tutorial project:
In Android Studio, select File > New > Import Project**.**
Go to the location where you saved the Google Maps Android API v2 Samples repository after downloading it.

Find the MapWithMarker project at this location:

PATH-TO-SAVED-REPO/android-samples/tutorials/MapWithMarker
Select the project directory, then click OK**.** Android Studio now builds your project, using the Gradle build tool.

Get an API key and enable the necessary APIs To complete this tutorial, you need a Google API key that's authorized to use the Maps SDK for Android.Click the button below to get a key and activate the API.

Add the API key to your app

1. Edit your project's gradle.properties file.

2. Paste your API key into the value of the GOOGLE_MAPS_API_KEY property. When you build your app, Gradle copies the API key into the app's Android manifest, as explained below.

GOOGLE_MAPS_API_KEY=PASTE-YOUR-API-KEY-HERE

Build and run your app
Connect an Android device to your computer. Follow the instructions to enable developer options on your Android device and configure your system to detect the device. (Alternatively, you can use the Android Virtual Device (AVD) Manager to configure a virtual device. When choosing an emulator, make sure you pick an image that includes the Google APIs.

In Android Studio, click the Run menu option (or the play button icon). Choose a device as prompted.Android Studio invokes Gradle to build the app, and then runs the app on the device or on the emulator. You should see a map with a marker pointing at Sydney on the east coast of Australia, similar to the image on this page.
Troubleshooting:
If you don't see a map, check that you've obtained an API key and added it to the app, as described above. Check the log in Android Studio's Android Monitor for error messages about the API key.
Use the Android Studio debugging tools to view logs and debug the app.

Understand the code
This part of the tutorial explains the most significant parts of the MapWithMarker app, to help you understand how to build a similar app.

Check your Android manifest
Note the following elements in your app's AndroidManifest.xml file:
Add a meta-data element to embed the version of Google Play services that the app was compiled with.

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

Add a meta-data element specifying your API key. The sample accompanying this tutorial maps the value for the API key to a string google_maps_key. When you build your app, Gradle copies the API key from your project's gradle.properties file to the string value.

```
<meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="@string/google_maps_key" />
```

To see how the API key maps to the string value, take a look at your app's build.gradle. It contains the following line that maps the string google_maps_key to the gradle property GOOGLE_MAPS_API_KEY:

```
  resValue "string", "google_maps_key",
          (project.findProperty("GOOGLE_MAPS_API_KEY") ?: "")
```

Below is an example of a full manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mapwithmarker">
```

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <!--
             The API key for Google Maps-based APIs.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsMarkerActivity"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Add a map

Display a map, using the Maps SDK for Android.

1. Add a <fragment> element to your activity's layout file, activity_maps.xml. This element defines aSupportMapFragment to act as a container for the map and to provide access to the GoogleMap object. The tutorial uses the Android support library version of the map fragment, to ensure backward compatibility with earlier versions of the Android framework.

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/map"
   android:name="com.google.android.gms.maps.SupportMapFragment"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.example.mapwithmarker.MapsMarkerActivity" />
```

In our activity's onCreate() method,we set the layout file as the content view. Get a handle to the map fragment by calling FragmentManager.findFragmentById(). Then use getMapAsync() to register for the map callback:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   // Retrieve the content view that renders the map.
   setContentView(R.layout.activity_maps);
   // Get the SupportMapFragment and request notification
   // when the map is ready to be used.
   SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
         .findFragmentById(R.id.map);
   mapFragment.getMapAsync(this);
}
```

Implement the OnMapReadyCallback interface and override the onMapReady() method, to set up the map when the GoogleMap object is available:

```
public class MapsMarkerActivity extends AppCompatActivity
      implements OnMapReadyCallback {
   // Include the OnCreate() method here too, as described above.
   @Override
   public void onMapReady(GoogleMap googleMap) {
```

```
        // Add a marker in Sydney, Australia,
        // and move the map's camera to the same location.
        LatLng sydney = new LatLng(-33.852, 151.211);
        googleMap.addMarker(new MarkerOptions().position(sydney)
                .title("Marker in Sydney"));
        googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

By default, the Maps SDK for Android displays the content of the info window when the user taps a marker. There's no need to add a click listener for the marker if you're happy to use the default behavior.

Congratulations! You've built an Android app that displays a Google map with a marker to indicate a particular location. You've also learned how to use the Maps SDK for Android.

Next steps

Learn more about the map object and what you can do with markers.

3.2.5 Scan QR-Code

Nowadays Barcodes and QR Codes are widely used in lot of mobile apps. In a QR Code you can store information like text, sms, email, url, image, audio and few other formats. In Android you can extract the information stored in barcodes by using Google Vision Library. Even though there are lot of other libraries available, google vision library is best to consider as it's not only provide barcode reading but also have other features like face detection, text detection.
In this part we are going to discuss how to use the google vision library by creating a simple movie ticket scanning app.

1. Google Mobile Vision API

Google Mobile Vision api helps in finding objects in an image or video. It provides functionalities like face detection**,** text detection and barcode detection. All these functionalities can be used separately or combined together.

This article aims to explain the barcode detection with a realtime use case scenario. We can see lot of barcode scanning apps used in supermarkets**,** theatres and hotels which scans a barcode and

provides user desired information. In this article we'll try to build a simple movie ticket scanner app which scans a barcode / qrcode and displays the movie information to book a ticket.
The google vision library is a part of play services and can be added to your project's bild.g
Complle 'com.google.android.gms:play-services-version:11.0.2'

2. Barcode Scanner Library

Google provided a simple tutorial to tryout the barcode scanning library with a simple bitmap image. But when it comes to scanning a realtime camera feed for a barcode, things become difficult to implement as we need to perform barcode detection on camera video.
I have developed a simple barcode scanner library by forking the google vision sample. In this library few bugs were fixed and added other functionalities like callbacks when barcode is scanned and a overlay scanning line indicator that can be used in your apps.

3. How to Use the Barcode library

Follow the below simple steps to include the barcode / qrcode library in your project.

1. Add the androidhive barcode reader and google vision library to your app's build.gradle file.

build.gradle

```
dependencies {
   // barcode reader library
   implementation 'info.androidhive:barcode-reader:1.1.5'

   // google vision library
   implementation 'com.google.android.gms:play-services-vision:11.0.2'
}
```

2. Add the barcode camera fragment to your activity or fragment.

```
<fragment
 android:id="@+id/barcode_scanner"
      android:name="info.androidhive.barcode.BarcodeReader"
      android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    app:auto_focus="true"
    app:use_flash="false" />
```

3. Implement your Activity from BarcodeReader.BarcodeReaderListener and override the necessary methods.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.SparseArray;
import com.google.android.gms.vision.barcode.Barcode;
import java.util.List;

import info.androidhive.barcode.BarcodeReader;

public class MainActivity extends AppCompatActivity implements
BarcodeReader.BarcodeReaderListener {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_scan);
   }

   @Override
   public void onScanned(Barcode barcode) {
      // single barcode scanned
   }

   @Override
   public void onScannedMultiple(List<Barcode> list) {
      // multiple barcodes scanned
   }

   @Override
   public void onBitmapScanned(SparseArray<Barcode> sparseArray) {
      // barcode scanned from bitmap image
   }
```

```
    @Override
    public void onScanError(String s) {
        // scan error
    }
  @Override
    public void onCameraPermissionDenied() {
        // camera permission denied
    }
}
```

4. Run your project and try to scan barcode or qrcode. The scanned result will be returned in onScanned() or onScannedMultiple() method.

Adding Scanning Overlay Indicator Line

We can see all the scanning apps generally adds an indicator line on the camera overlay to indicate the scanning progress in going on. To achieve this, I have added a reusable class in the same library which can be added on to camera screen.

To add the animating scanning line, add the info.androidhive.barcode.ScannerOverlay to same activity overlapping the camera fragment.

```
<info.androidhive.barcode.ScannerOverlay
    android:layout_width="match_parent"
    android:layout_height="match_parent"
     android:background="#44000000"
     app:line_color="#7323DC"
     app:line_speed="6"
     app:line_width="4"
     app:square_height="200"
     app:square_width="200"/>
```

Chapter-4
Technical Description

4.1 Introduction

The chapter describes System blocks, characteristics, working principle of microcontroller Arduino UNO, relay, keypad, DC lock system, potentiometer of our project. A microcontroller is a basically a little PC (SoC) on a solitary coordinated circuit containing a processor center, memory, and programmable information/yield peripherals. This is the reason for the simple to-computerized converter (ADC). Arduino is an open-source PC equipment and programming organization, task and client group that outlines and produces microcontroller-based units for building advanced gadgets and intuitive articles that can sense and control objects in the physical world. Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on a buzzer, publishing something online.

4.2 Overview of the total system

Whenever the keys are pressed, they are matched with the keys already stored. If the keys that are pressed match the initial password stored in the EEPROM, then the lock will open.. If the password does not match for three times, then it will print “access denied” on the LCD display and a buzzer will also ring. If the ‘#’ key is pressed, it will ask the user to enter the current password and if it matches, then it will ask you to enter the new password and the password will be changed. Keypad is used for taking password. Buzzer is used for indications.

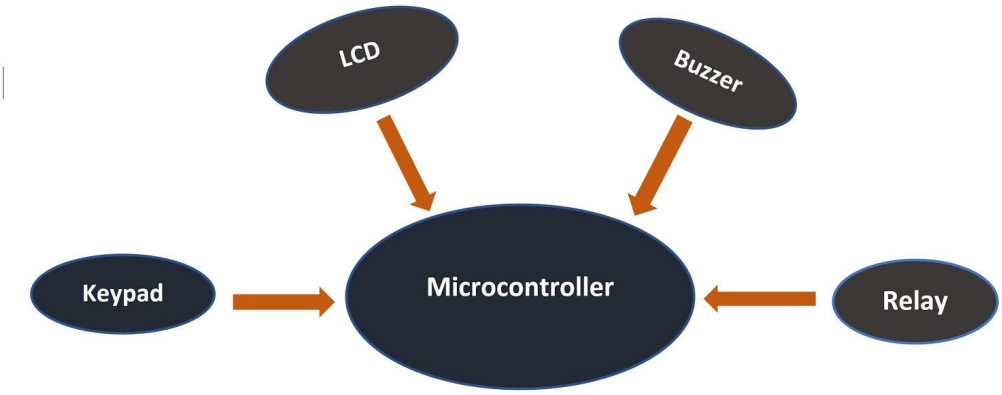


Fig. 4.1 Simplified block diagram

4.3 System Blocks

In this section total block diagram of the system including all the subsystems will be shown.

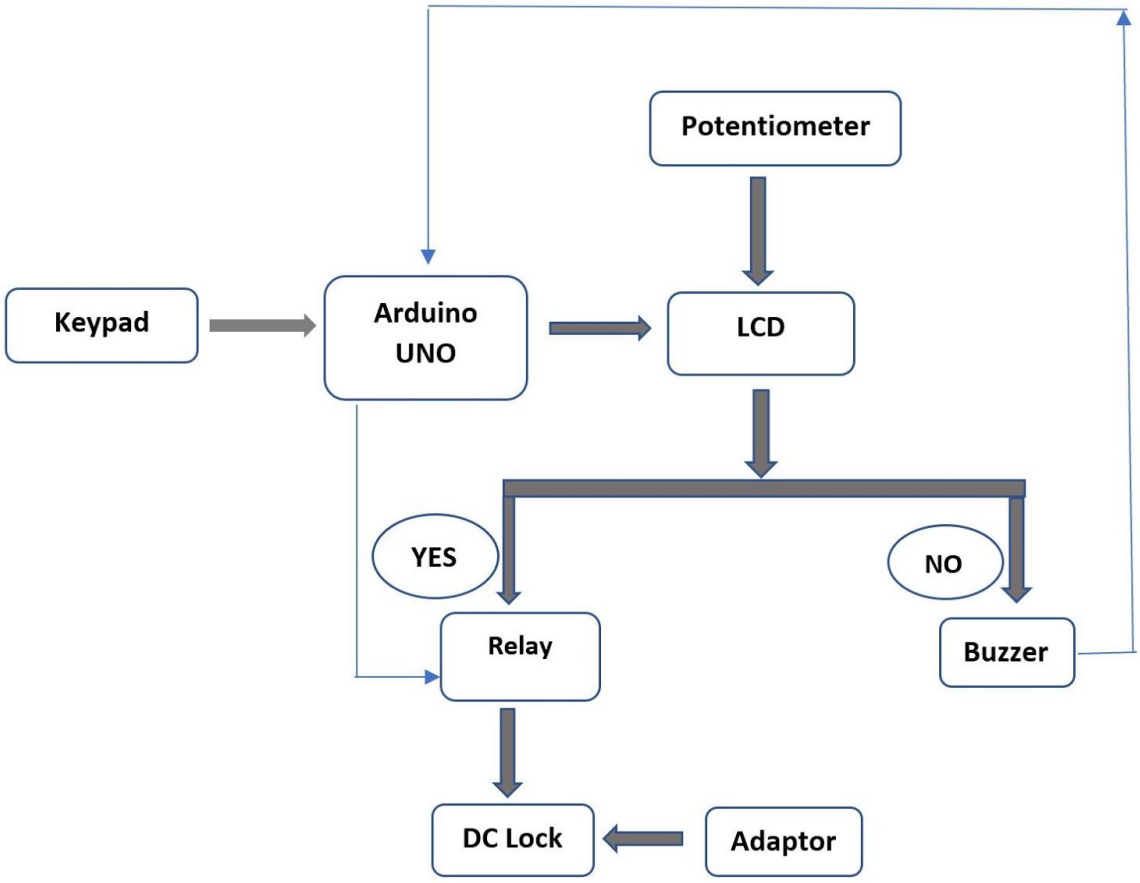


Fig. 4.2 Detailed block diagram of the whole system.

Microcontrollers is used to build the designed robot. We starts with a central microcontroller which we will use to keep communication and for synchronization with all the subsystems. Arduino is an open source microcontroller-based hardware platform which is used the system. It is very easy to work with various kinds of sensors to interact with the external physical world. Wikipedia states “Arduino is a single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open-source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.”[16]

Here we are using Arduino Uno ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It features the Atmega16U2 programmed as a USB-to-serial converter. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

4.4 Relay Switch

A relay is an electrically operated switch. Usually relays use an electromagnet to mechanically trigger a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuitry by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. Relays are switches that open and close circuits electro mechanically or electronically. Relays control one electrical circuit by opening and closing contacts in another circuit. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized. When a relay contact is Normally Closed (NC), there is a closed contact when the relay is not energized. In either case, applying electrical current to the contacts will change their state. Relays are generally used to switch smaller currents in a control circuit and do not usually control power consuming devices except for small motors and Solenoids that draw low amps. Nonetheless, relays can "control" larger voltages and amperes by having an amplifying effect because a small voltage applied to a relays coil can result in a large voltage being switched by the contacts. Protective relays can prevent equipment damage by detecting electrical abnormalities, including overcurrent, undercurrent, overloads and reverse currents. In addition, relays are also widely used to switch starting coils, heating elements, pilot lights and audible alarms.[19]

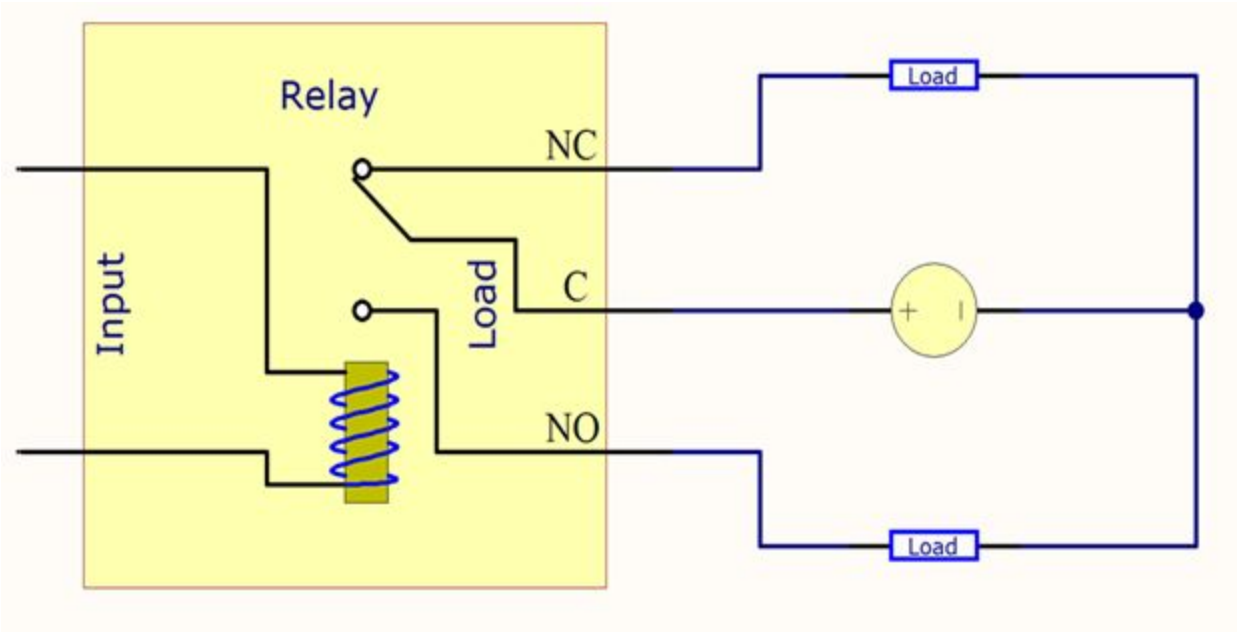


Fig. 4.3 Internal diagram of a relay

4.5 Keypad (4*4)

This 4x4 matrix keypad has 16 built-in push button contacts connected to row and column lines. So, the program has to tell it our keypad is has 4 rows and 4 columns, which I/O pins the lines are connected to, and what value each button represents. The rows, cols, and values arrays store that information. There are numerous techniques depending on the connection keypad with microcontroller, but the fundamental logic is same the columns are made as input and drive the rows making them as output. So as to detect which key is pressed from the matrix keypad, the row lines are to be made low one by one and read the columns. 4x4 Matrix Keypad Interfacing with 8051 Microcontroller. Keypads are widely used input devices being used in various electronics and embedded projects. They are used to take inputs in the form of numbers and alphabets, and feed the same into system for further processing.

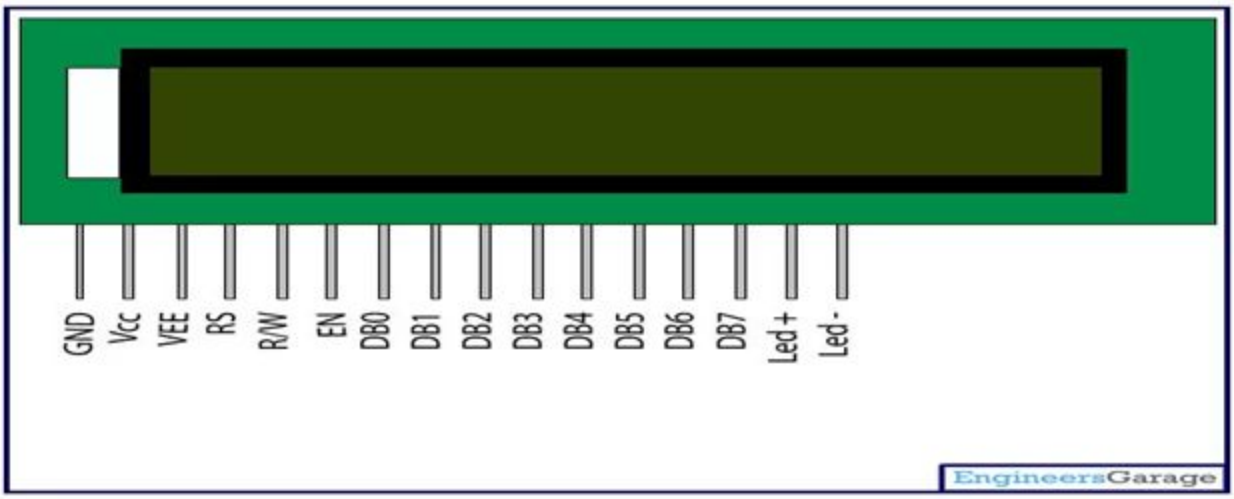
4.6 16*2 LCD (Liquid Crystal Display)

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

4.7 Pin Diagram



4.8 Pin Description

Pin No	Function	Name
1	Ground (0V)	GND
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	VEE
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	R/w
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3

11	8-bit data pins	DB4
12		DB5
13		DB6
14		DB7
15	Backlight VCC (5V)	Led+
16	Backlight V_{CC} (5V)	Led-

4.9 Adapter

An adapter or adaptor[1] is a device that converts attributes of one device or system to those of an otherwise incompatible device or system. Some modify power or signal attributes, while others merely adapt the physical form of one connector to another. An AC-to-DC power supply adapts electricity from household mains voltage (either 120 or 230 volts AC) to low-voltage DC suitable for powering consumer electronics. Small, detached power supplies for consumer electronics are called AC adapters, or variously power bricks, wall warts, or chargers.

- Input Voltage: 100~240V AC, 50/60Hz
- Output Voltage: 12V DC, 2000mA
- Plug: US plug
- Cable length: 100cm
- Net weight: 128g

4.10 Buzzer

This one generates a continuous beep usually when supplied with power but you can generate any tone as you wish by interfacing it with a microcontroller with proper coding.
Type: Magnetic Transducer, Self-Drive (External Drive with Feedback)
Sound Pressure Level: 82dB
Mounting Type: Through Hole
Frequency: 2.3kHz
Packaging: Tray

Operating Mode: Continuous
Voltage Range: 2 ~ 5VDC
Voltage - Rated: 3VDC
Current Rating: 30mA
Size / Dimension: Circular - 12mm Dia x 7.5mm H
Termination: Solder Pins/Posts
Lead-Free Status: Lead Free
RoHS Status: RoHS Compliant

4.11 Power Bank

A power bank was used to power up the Arduino as well as Ethernet shield. All the equipment's was powered up by onboard power source without DC Lock.

Chapter 5
Hardware Implementation

5.1 Introduction

The chapter gives us clear idea and view about the implementation and the steps we have taken for hardware and software implementation. For hardware implementation there are some specific hardware to build up cycle security. The internal circuitry is built by hardware and developed by Arduino programming language. For the software part, we build an Android APP by the help of android studio. The whole procedure starting from building app to running codes and connecting hardware and software is done and discussed to take us near to the project.

5.2 List of necessary hardware

In this section each software will be discussed in details to clearly to understand the design as well as the software implementation process.
Required tools for Design implementation are-

- Arduino UNO
- Buzzer
- Keypad 4*4
- LCD16*2
- Relay
- 10k Potentiometer
- Adapter/Battery
- Power Bank
- Dc Lock
- Bread Board
- Wire

5.3 Working Procedure

In this section, each software will be discussed in details to clearly understand the design as well as the software implementation process. After that, connect the LCD to the Arduino. The connections for connecting the LCD with the Arduino are as follows:

- Connect pin 1 on the LCD, which is the VSS pin, to GND on the Arduino
- Connect pin 2, which is the VDD pin, to the 5V pin on the Arduino
- Connect pin 3, which is the V0, to the middle of the 10k potentiometer and connect the other two pins on the potentiometer to 5V and GND on the Arduino. This pin is for setting the LCD's contrast.
- Connect pin 4, which is the RS pin, to pin 7 on the Arduino
- Connect pin 5, which is the R/W pin, to the GND pin on the Arduino
- Connect pin 6, which is the Enable pin, to pin 6 on the Arduino

- Connect pins 11, 12, 13, and 14 which are the data pins, to the pins 5, 4, 3, and 2 on the Arduino
- Connect pin 15, which is the LCD's backlight pin, to 5V on the Arduino through the 220-ohm resistor
- Connect pin 16 on the Arduino, which is the negative pin of the backlight, to GND on the Arduino

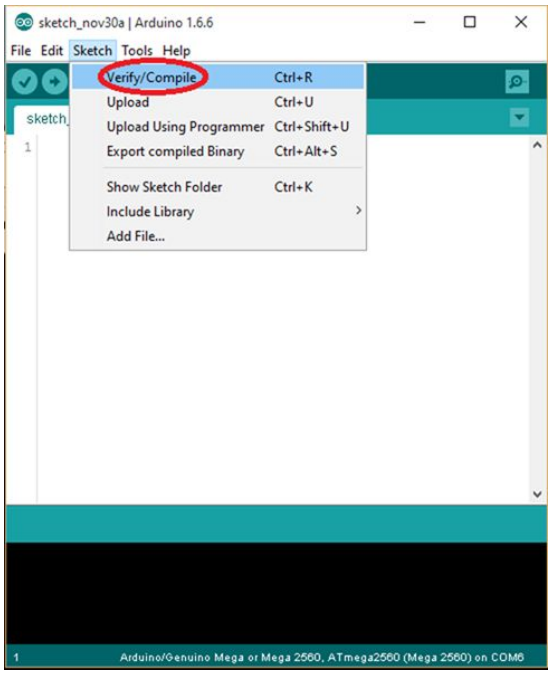
5.4 Programming the central Microcontroller

As a central microcontroller we are using ATMEGA 2560 which is based on an Arduino platform. Arduino uses a C language-based programming language called “Wiring”. All the programming codes have to be written on the Arduino IDE. There is a text editor in the Arduino IDE. This text editor is used to write the programming codes on it. There are two main parts of a typical Arduino program. One is “void setup()” and the other is “void loop()”. “void setup()” is a function that runs once at the start of a program and that can initialize settings. “void loop()” is a function called repeatedly until the board powers off.

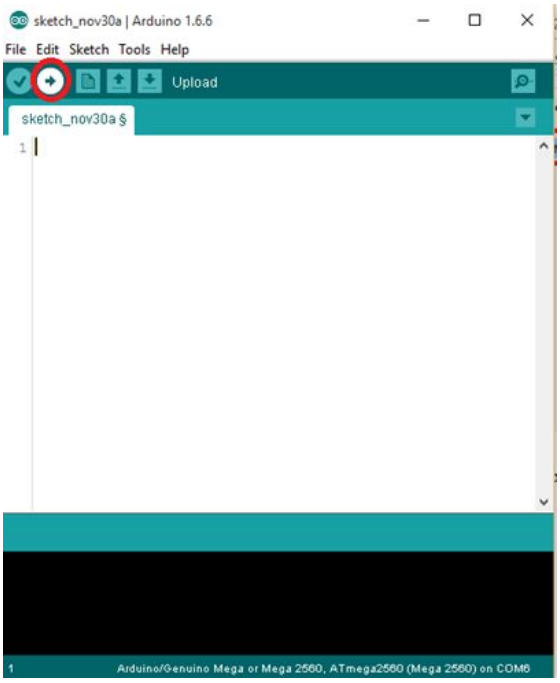
To declare a pin of the Arduino as a input “pinMode(pin number, INPUT)” is used. An to declare as a output “pinMode(pin number, OUTPUT)” is used. To get a digital data on an output pin “digitalWrite(pin number)” command is used. If we want to read a value from a sensor which is connected to an analog pin we use “analogRead(pin number)”. If we have some conditional situation we can use “if” conditional statement. We have used all these things to write program for the Arduino.

There are many libraries for the Arduino. The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. We have used libraries for Keypad, LCD, Buzzer, Relay, DC Lock.

After writing the whole code on the Arduino IDE editor, we compiled it through the compiler. To do this we select “Sketch” file menu on the Arduino window. Then we select “Verify/compile”. This will compile the code. If there is any error it will show that.



After compiling it we have to upload the program to the Arduino. First we need to connect the Arduino board to the USB port through an USB cable provided with the Arduino. Then we need to select "Upload" at the top left corner on the Arduino IDE. If the program is uploaded successfully it will show it through a message. Now the central microcontroller is programs. The whole programming code of the rover is in APPENDIX A.



Chapter 6
Design Impact

6.1 Economic Impact

Our BDRIDER app is produced low cost. For the sharing the bicycle owner is getting money and is going to be low cost for the user. This app is built in such a way that common people can afford it easily and use it for different purposes. The whole system is low costing which makes it cost efficient. By making market price low and produce more products it can benefit and serve economical purposes as well.

6.2 Environmental Impact

It down on greenhouse gas emissions and global climate change. Reduces air pollutants (walking and biking emit no greenhouse gases). Reduces noise pollution and congestion. Reduces the need for new parking lots and roadways. Save valuable green space from development. Reduces your ecological footprint.

6.3 Social / Safety Impact

Increased contact with your neighbors and community. Calmer and safer roads. More "eyes on the street." "Safety in numbers." It can also save time by reducing traffic jam.

6.4 Political Impact

Political leaders can be innovations of new ideas and new concepts. They can introduce new things to their people to reduce the burden of them by implementing new systems.

6.5 Ethical Impact

This app has no ethical dilemma rather by saving times and money. You will use BDRIDER only
for your personal use. We guarantee that your personal information will be fully protected
by us.

6.6 Health and Safety Impact

Cycling can better your health and well-being by:

- Reducing depression
- Reducing obesity
- Reducing adult-onset diabetes
- Reducing heart disease
- Reducing high blood pressure
- Reducing osteoporosis

Physical activity reduces the helth risks by:

- Improving posture/balance
- Lowering blood pressure
- Increasing energy
- Increasing flexibility and muscle strength
- Lowering stress levels

6.7 Sustainability

Our app has security, it is easy to use, saving time and energy.

6.8 Total Cost for Implementation

Here is our total cost for implementation of the project. We tried our best to reduce the cost as much as possible. We think we are successful to implement the rover within a limited cost.

Name of component	**Cost (Taka)**
Arduino Uno	400
Keypad 4*4	90
LCD 16*2	249
Relay	88
Battery	30
Power bank	800
Dc lock	40

Bread Board	70
Buzzer	15
10k potentiometer	10
Wire	40
Total	**1832**

Table: Cost of the components.

6.9 Result

The main objective behind of this App is that the students or tourists can easily ride the bicycle rent at a low cost. For this reason, we tried to keep a garage at main point and arranged the lock through the pass code. This App can be installed by using a any user from the play store. This app would be show nearby garage with an available bicycle.

From the pass code we can easily unlock the cycle and we also see that cycle data. We can easily use this app by Login or Register.

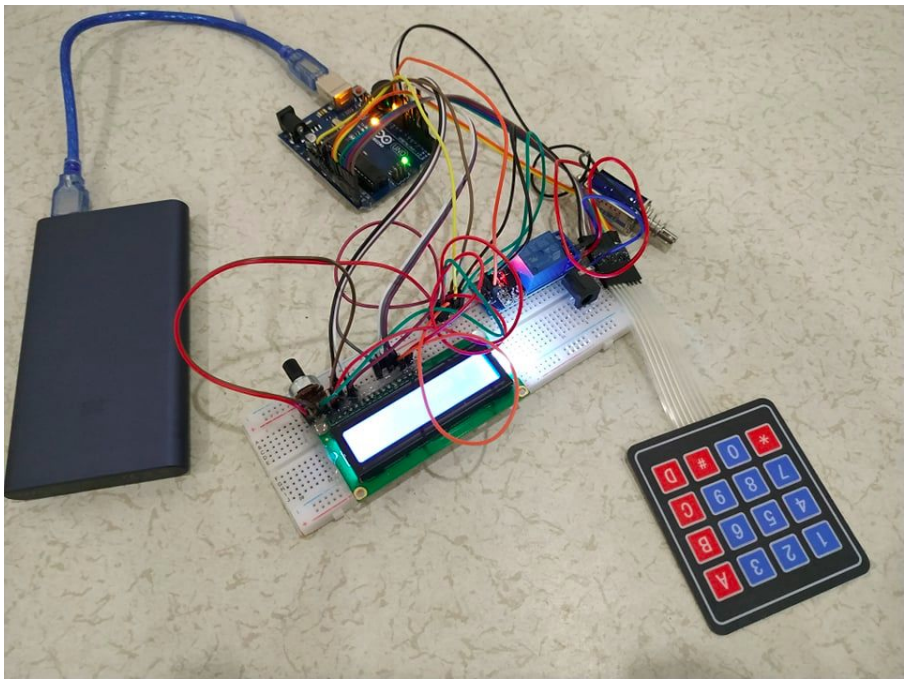


Fig. 8.1 Final picture internal of the lock system

In this project, we have successfully implemented the bicycle renting APP which can be very useful and effective in many students, tourists and other person.

Conclusion

In this project we implemented a cost-effective bicycle renting APP that can easily be used bicycle owners and users through their NID card information or registered mobile number. With the help of this APP, search and find the cycle can be made much effective and easier. For this APP owner can earn money from sharing the cycle and user save their money and time for renting the cycle. No one can else open the bicycle lock for this pass code. This project can be improved by using Bluetooth Module and tracker. In future we use the tracker for every cycle so that we can easily track the cycle.

We have achieved our goal of implementing a cost-effective cycle sharing application system which can detect an available cycle and nearby garage by monitoring the location gases so that it can help find the cycle where cannot go and search for renting cycle. Multiple ways were found to implement the project but the best and effective method was chosen, where the main objective was to make a better and useful APP within a limited cost.

Bibliography

[1]https://create.arduino.cc/projecthub/diy-hacking/arduino-keyless-door-lock-system-with-keypad-and-lcd-bcad2e
[2]https://create.arduino.cc/projecthub/ianabcumming/a-simple-arduino-menu-with-an-lcd-254080?ref=tag&ref_id=display&offset=12
[3] https://www.instructables.com/id/Arduino-Keypad-4x4-Tutorial/
[4] Available: https://en.wikipedia.org/wiki/Arduino
[5] Available: https://en.wikipedia.org/wiki/Relay
[6] http://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/
[7] https://www.arduinouno.cc/
[8] https://www.circuito.io/blog/arduino-uno-pinout/
[9https://www.google.com/search?client=ubuntu&channel=fs&q=firebase+console&ie=utf-8&oe=utf-8
[10] https://console.firebase.google.com/u/0/
[11] https://console.firebase.google.com/u/0/project/myfirebasedatabase-fc388/overview

Appendices

Appendix A

Locking System Aurino Code

Appendix A
Lock Code:
[code]

```
#include <Keypad.h>
#include<LiquidCrystal.h>
#include<EEPROM.h>
#include <SoftReset.h>
LiquidCrystal liquid_crystal_display(2,3,4,5,6,7);
int passcount=0;
char password[4];
int buzzer=11;
char initial_password[4],new_password[4];
int i=0;
int relay_pin = 10;
char key_pressed=0;
const byte rows = 4;
const byte columns = 4;
char hexaKeys[rows][columns] = {

{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};

byte row_pins[rows] = {A0,A1,A2,A3};
byte column_pins[columns] = {A4,A5,9,8};

Keypad keypad_key = Keypad( makeKeymap(hexaKeys), row_pins,

column_pins, rows, columns);

void setup()
{
  pinMode(relay_pin, OUTPUT);
```

```
   pinMode (buzzer, OUTPUT);
   digitalWrite(relay_pin, HIGH);

  liquid_crystal_display. Begin(16,2);
  liquid_crystal_display.print(" 499 Project ");
  liquid_crystal_display.setCursor(0,1);
  liquid_crystal_display.print("Electronic Lock ");

  delay(2000);

  liquid_crystal_display.clear();
  liquid_crystal_display.print("Enter Password");
  liquid_crystal_display.setCursor(0,1);
 // intialpassword();

}
void loop()
{

  digitalWrite(relay_pin, HIGH);
 key_pressed = keypad_key.getKey();

  if(key_pressed=='#')
    change();
  if (key_pressed)
  {
    password[i++]=key_pressed;

    liquid_crystal_display.print(key_pressed);
   }
  if(i==4)
  {
   delay(200);
for(int j=0;j<4;j++)
initial_password[j]=EEPROM.read(j);

if(!(strncmp(password, initial_password,4)))
   {
      passcount=0;
```

```
     liquid_crystal_display.clear();
    liquid_crystal_display.print("Pass Accepted");
   digitalWrite(relay_pin, LOW);
   delay(2000);

  liquid_crystal_display.setCursor(0,1);   liquid_crystal_display.print("Pres # to change");

     delay(2000);
       soft_restart();
     liquid_crystal_display.clear();
     liquid_crystal_display.print("Enter Password:");
      liquid_crystal_display.setCursor(0,1);

     i=0;
  }
 else
 {
   digitalWrite(relay_pin, HIGH);
passcount++;


     liquid_crystal_display.clear()
     liquid_crystal_display.print("Wrong Password");
     liquid_crystal_display.setCursor(0,1)
     liquid_crystal_display.print("Pres # to Change");

     delay(2000);

     liquid_crystal_display.clear()
     liquid_crystal_display.print("Enter Password");
     liquid_crystal_display.setCursor(0,1);
     i=0;
   }

if(passcount>2){
  digitalWrite(buzzer,HIGH);
  delay(100);
  digitalWrite(buzzer, LOW);
   delay(100);
```

```
   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);

   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
      digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);

   digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
  delay(100);
   digitalWrite(buzzer,HIGH);
```

```
  delay(100);
  digitalWrite(buzzer,LOW);
   delay(100);
        digitalWrite(buzzer,HIGH);
  delay(100);
  digitalWrite(buzzer,LOW);
   delay(100);
     digitalWrite(buzzer,HIGH);
  delay(100);
  digitalWrite(buzzer,LOW);
   delay(100);

     digitalWrite(buzzer,HIGH);
  delay(100);
  digitalWrite(buzzer,LOW);
   delay(100);
     digitalWrite(buzzer,HIGH);
  delay(100);
  digitalWrite(buzzer,LOW);
   delay(100);
}
  }
}
void change()
{
int j=0;

  liquid_crystal_display.clear();
  liquid_crystal_display.print("Current Password");
  liquid_crystal_display.setCursor(0,1);

  while(j<4)
  {
   char key=keypad_key.getKey();
   if(key)
   {
    new_password[j++]=key;

    liquid_crystal_display.print(key)
```

```
}
  key=0;
 }
 delay(500);
 if((strncmp(new_password, initial_password, 4)))

 {
  liquid_crystal_display.clear()
  liquid_crystal_display.print("Wrong Password");
  liquid_crystal_display.setCursor(0,1);
  liquid_crystal_display.print("Try Again");

  delay(1000);

 }
 else
 {
  j=0;

  liquid_crystal_display.clear();
  liquid_crystal_display.print("New Password:");
  liquid_crystal_display.setCursor(0,1);
  while(j<4)
  {
   char key=keypad_key.getKey();
   if(key)

     initial_password[j]=key;
     liquid_crystal_display.print(key);
     EEPROM.write(j,key);
   j++;
 }
  }
 liquid_crystal_display.print("Pass Changed");

  delay(1000);
 }

 liquid_crystal_display.clear();
```

```
  liquid_crystal_display.print("Enter Password");
  liquid_crystal_display.setCursor(0,1);

  key_pressed=0;
}
void initialpassword (){
  for(int j=0;j<4;j++)
   EEPROM.write(j, j+49);
 for(int j=0;j<4;j++)
    initial_password[j]=EEPROM.read(j);
}
```

Appendix B

Construction for the USER

- First installed our app BDRIDER from the play store
- Turn on the WIFI or connect the internet on the device and set the location\
- If you have already been registered then you can login our app otherwise you have to open an account through your NID information or Mobile number.
- Now you can search nearest garage
- In the garage have QR code you have to scan this code
- Then when you scan the cycle QR code you get a code which is help for the unlock the cycle and all about the cycle information

Now you can enjoy your ride.