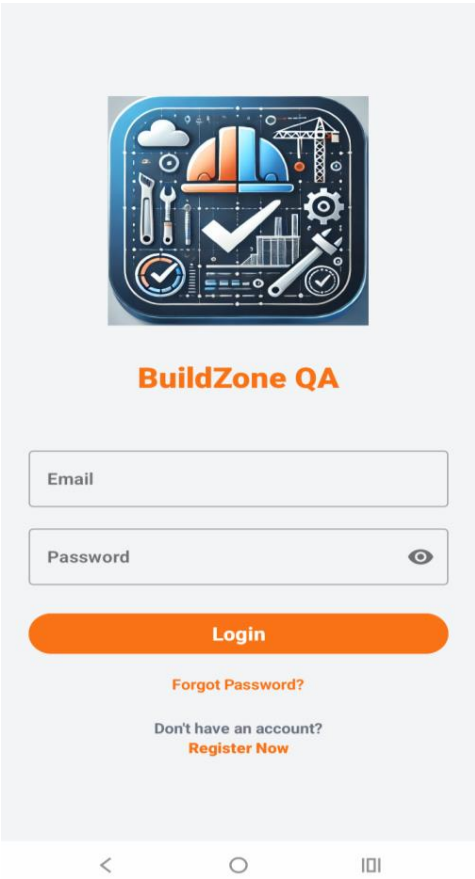
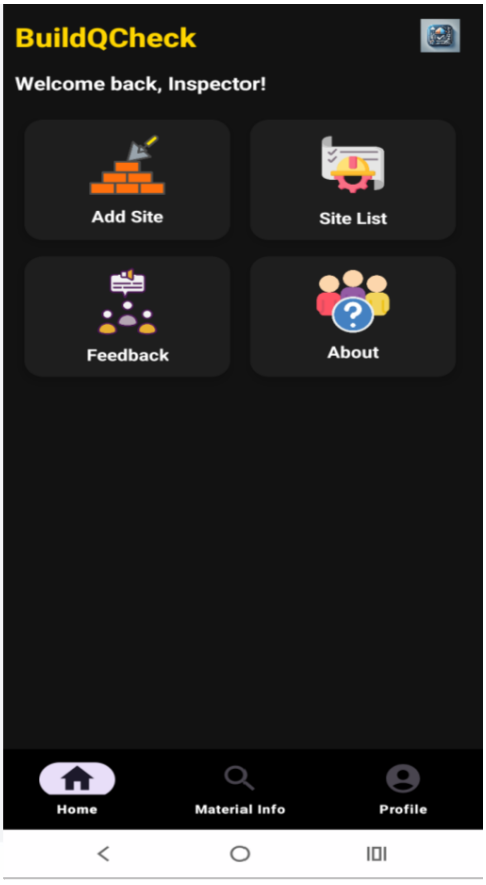
Introduction – BuildQcheck

The construction quality monitoring application is built around a main HomeScreen, which serves as the starting point for accessing all key features of the app.

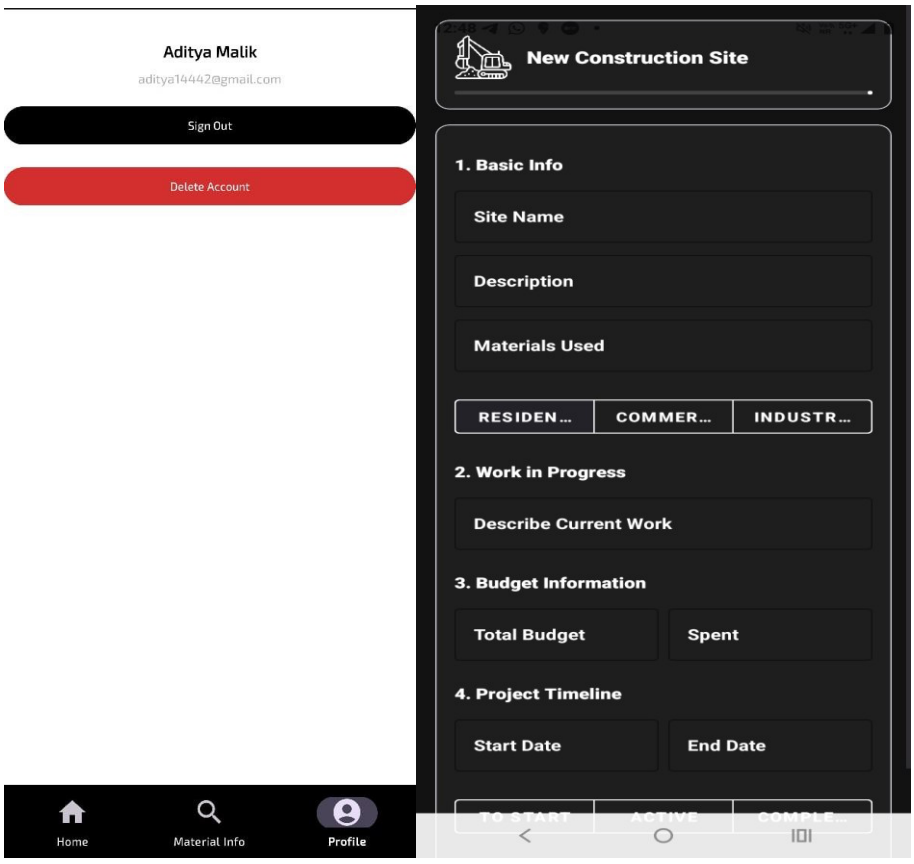
- The SiteListScreen shows a list of ongoing construction projects. From here, users can tap on any site to open the SiteDetailsScreen, where they can explore specific options related to that site:
    - The CheckQualityScreen allows field workers to carry out real-time quality inspections.
    - The ReportIssueScreen lets users record any problems or safety concerns they notice on-site.
    - The ViewReportsScreen gives users the ability to look back at previous inspections and reports for reference and tracking.
- To register a new construction site, users can use the AddSiteScreen.
- The ProfileScreen provides access to personal user data, settings, and the option to log out.

Login page and Home page

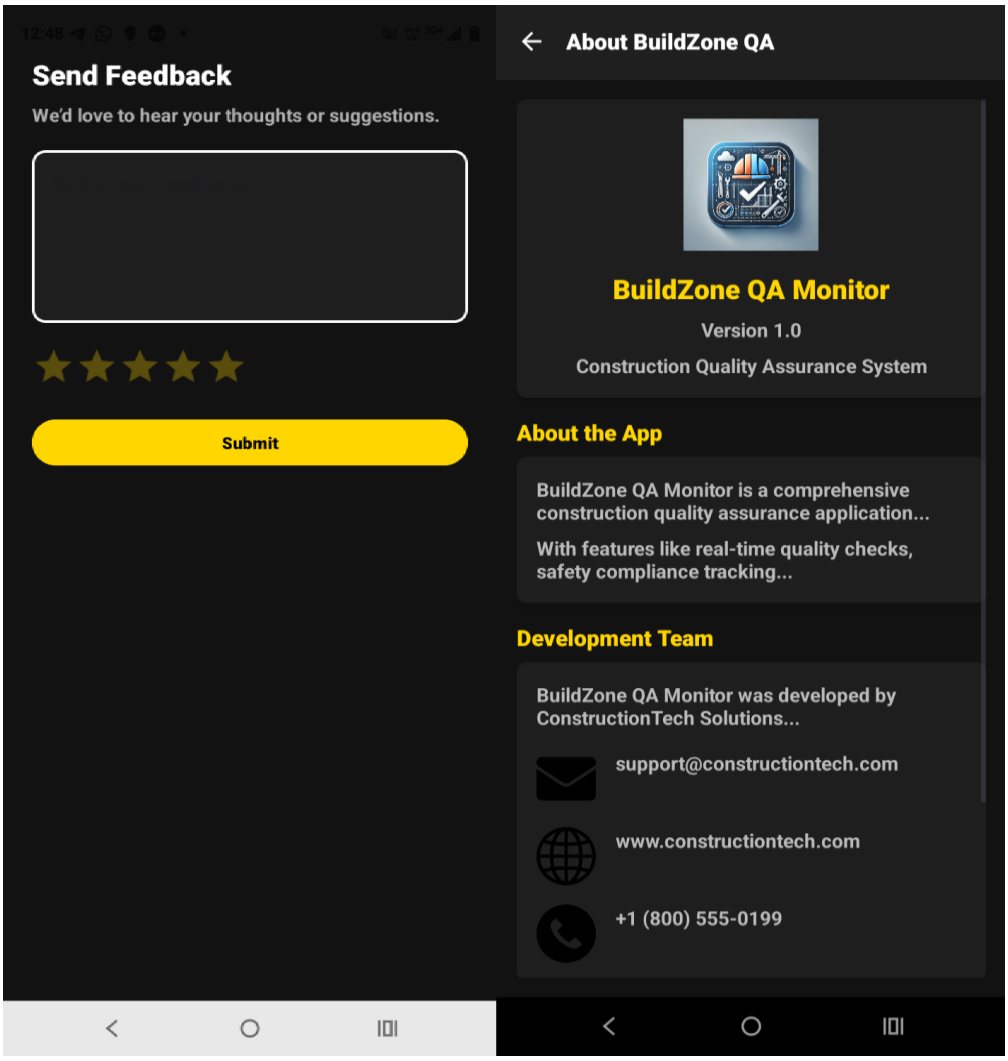




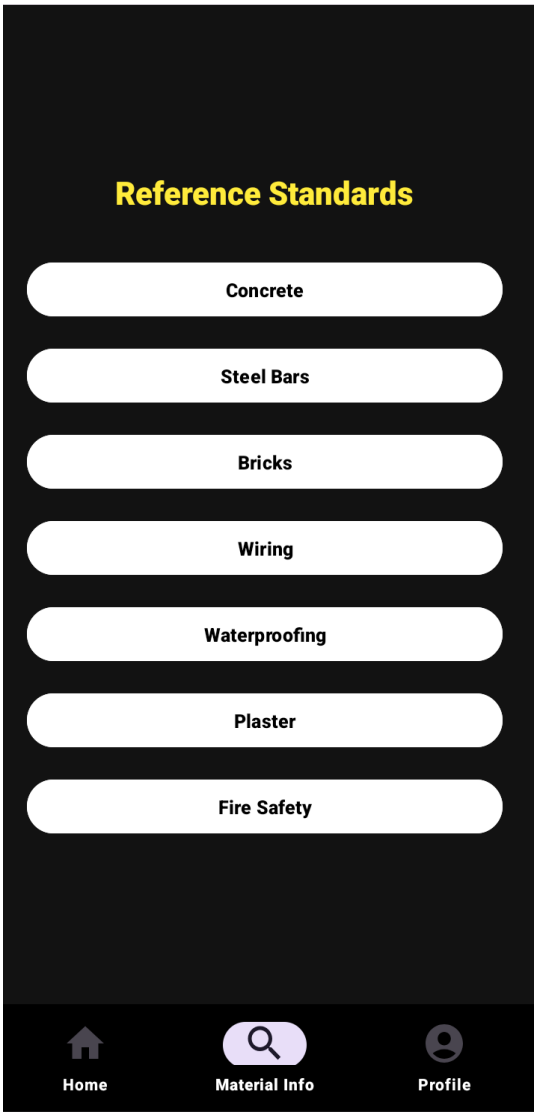
Profile and Add Site

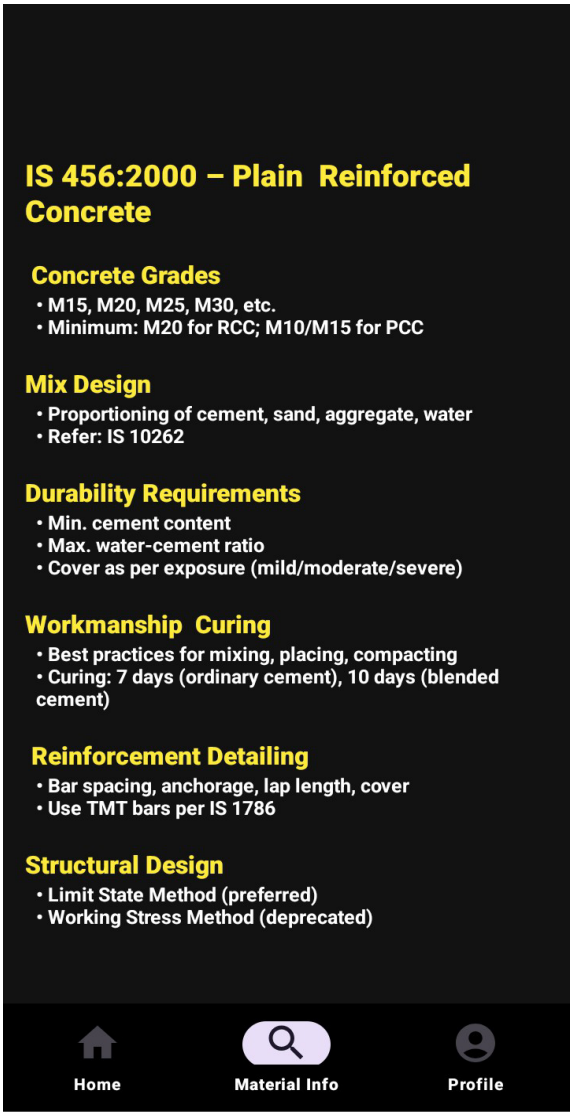


Feedback and About App



Materials Info





Material Info and Sitelist

NBC 2016 – Part 4: Fire and Life Safety

1. Fire Zones
- Classified into Fire Zones I, II, III based on risk
- Construction materials & types must suit zone category

2. Building Occupancy Classification
- A – Residential
- B – Educational
- C – Institutional
- D – Assembly
- E – Business
- F – Industrial
- H – Storage
- J – Hazardous

3. Fire Protection Systems
- Fire extinguishers, hydrants, wet risers
- Sprinklers, smoke detectors, alarm systems
- Required based on height & usage

4. Exit Requirements
- Minimum number of exits per floor
- Stair & door size, travel distance guidelines
- Fire escape stairs must be fire-rated
- Emergency lighting is mandatory

5. Fire Resistance of Materials
- Fire rating for walls, floors, doors, structures
- Fire-retardant doors: 120 min minimum

6. Testing and Maintenance
- Regular fire drills & audits
- Testing of pumps, alarms, sprinklers

7. Fire Command Centre (High-rises)
- Required for buildings >15m height
- Control panel, CCTV, fire system monitoring

Test

test

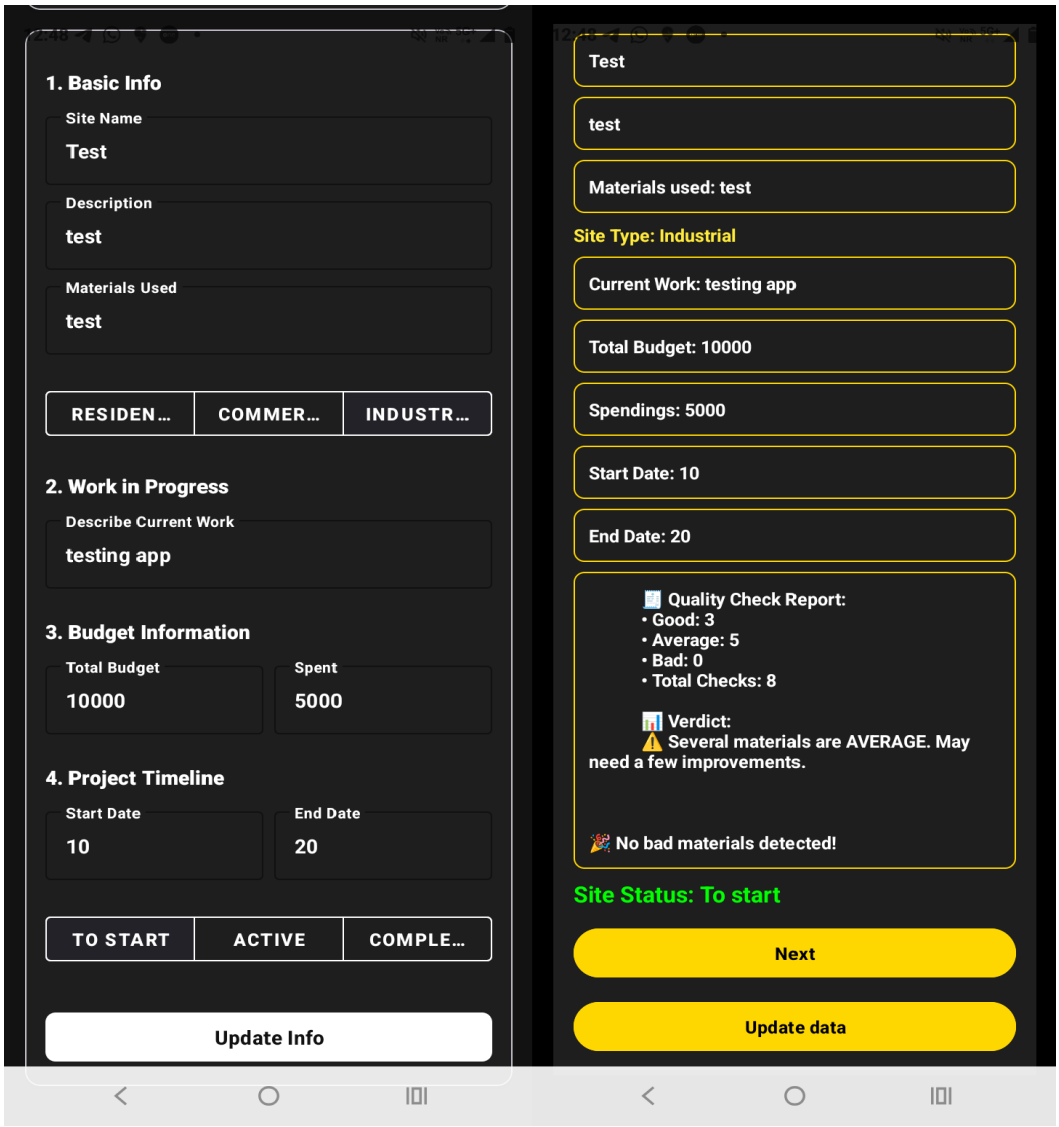
Status: To start

Witch's Hut

A house for the witch

Status: Active

Quality Check



SIGN UP

```java
package com.example.bqc

import ...

class SignUp : AppCompatActivity() {
    private lateinit var binding: ActivitySignUpBinding
    private lateinit var firebaseAuth : FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        firebaseAuth = FirebaseAuth.getInstance()

        binding = ActivitySignUpBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnSignup.setOnClickListener{
            val name = binding.etFullname.text.toString()
            val email = binding.etEmail.text.toString()
            val password = binding.etPassword.text.toString()
            val confPass = binding.etConfirmPassword.text.toString()

            if (name.isNotEmpty() && email.isNotEmpty() && password.isNotEmpty() && confPass.isNotEmpty()){
                if (password == confPass){
                    firebaseAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener{
                        if(it.isSuccessful){
                            val profileUpdates = userProfileChangeRequest {
                                displayName = name
                            }
                            firebaseAuth.currentUser?.updateProfile(profileUpdates)
                            val user = firebaseAuth.currentUser
                            user?.sendEmailVerification()?.addOnCompleteListener{
                                firebaseAuth.signOut()
                                if(it.isSuccessful){
                                    Toast.makeText( context: this, text: "Check your mail for verification link.", Toast.LENGTH_LONG).show()
                                }
                                else{
                                    Toast.makeText( context: this,it.exception.toString(), Toast.LENGTH_SHORT).show()
                                }
                            }
                        }
                        else{
                            Toast.makeText( context: this, text: "${it.exception.toString()}", Toast.LENGTH_LONG).show()
                        }
```

```
                        }
                        else{
                            Toast.makeText( context: this, text: "${it.exception.toString()}", Toast.LENGTH_LONG).show()
                        }
                    }
                }
                else{
                    Toast.makeText( context: this, text: "Passwords do not match.", Toast.LENGTH_SHORT).show()
                }
            }
            else{
                Toast.makeText( context: this, text: "One or more field is empty.", Toast.LENGTH_SHORT).show()
            }
        }




        binding.tvBackToSignin.setOnClickListener{
            val i = Intent( packageContext: this,SignIn::class.java)
            startActivity(i)
        }

    }

}
```

Sign in

```
package com.example.bqc

import ...

class SignIn : AppCompatActivity() {
    private lateinit var binding: ActivitySignInBinding
    private lateinit var firebaseAuth : FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        firebaseAuth =  FirebaseAuth.getInstance()

        binding = ActivitySignInBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.tvForgotPassword.setOnClickListener{
            val i = Intent( packageContext: this,ForgotPassword::class.java)
            startActivity(i)
        }
        binding.btnSignin.setOnClickListener{
            val email = binding.etEmail.text.toString()
            val password = binding.etPassword.text.toString()
            if(email.isNotEmpty() && password.isNotEmpty()){
                firebaseAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener{
                    if(it.isSuccessful){
                        val user=firebaseAuth.currentUser
                        if(user!=null) {
                            if (user.isEmailVerified) {
                                val i = Intent( packageContext: this, MainActivity::class.java)
                                startActivity(i)
                            } else {
                                Toast.makeText(
                                    context: this,
                                    text: "Your email is not verified. Check your mail for verification link",
                                    Toast.LENGTH_LONG
                                ).show()
                            }
                        }
                    }
                    else{
                        Toast.makeText( context: this,it.exception.toString(), Toast.LENGTH_SHORT).show()
                    }
                }
```

```
                    }
                }
            }
            else{
                Toast.makeText( context: this, text: "One or more fields is empty.", Toast.LENGTH_SHORT).show()
            }
        }


        binding.tvSignup.setOnClickListener{
            val i = Intent( packageContext: this,SignUp::class.java)
            startActivity(i)
        }
    }
    override fun onStart() {
        super.onStart()
        firebaseAuth = FirebaseAuth.getInstance()
        val user = firebaseAuth.currentUser
        if(user!=null && user.isEmailVerified){
            val i = Intent( packageContext: this,MainActivity::class.java)
            startActivity(i)
        }
    }
}
```

Forgot Password

```
package com.example.bqc

import ...

class ForgotPassword : AppCompatActivity() {
    private lateinit var firebaseAuth : FirebaseAuth
    private lateinit var binding : ActivityForgotPasswordBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityForgotPasswordBinding.inflate(layoutInflater)
        setContentView(binding.root)
        firebaseAuth = FirebaseAuth.getInstance()
        binding.btnResetpassword.setOnClickListener {
            val email = binding.etEmail.text.toString().trim()
            if (email.isNotEmpty()){
                firebaseAuth.sendPasswordResetEmail(email).addOnCompleteListener(this){task->
                    if (task.isSuccessful){
                        Toast.makeText( context: this, text: "Check your mail for password reset link.", Toast.LENGTH_LONG).show()
                        val i = Intent( packageContext: this,SignIn::class.java)
                        startActivity(i)
                        finish()
                    }
                    else{
                        Toast.makeText( context: this,task.exception.toString(), Toast.LENGTH_SHORT).show()
                    }
                }
            }
            else{
                Toast.makeText( context: this, text: "Email cannot be empty", Toast.LENGTH_SHORT).show()
            }
        }

    }
}
```

Splash Screen

```
package com.example.bqc

import ...

class SplashScreen : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_splash_screen)
        android.os.Handler().postDelayed({
            val i = Intent( packageContext: this,SignIn::class.java)
            startActivity(i)
            finish()
        }, delayMillis: 1000)
    }
}
```

About Screen

```
package com.example.bqc

import ...

class AboutScreen : AppCompatActivity() {
    private lateinit var binding: ActivityAboutScreenBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityAboutScreenBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setSupportActionBar(binding.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        supportActionBar?.title = ""
        val versionName = packageManager.getPackageInfo(packageName, 0).versionName
        binding.txtVersion.text = "Version $versionName"
    }

    fun openPrivacyPolicy(view: View) {
        val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.constructiontech.com/privacy"))
        startActivity(intent)
    }

    fun openTermsOfService(view: View) {
        val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.constructiontech.com/terms"))
        startActivity(intent)
    }

    override fun onSupportNavigateUp(): Boolean {
        finish()
        return true
    }
}
```

Add Site

```java
package com.example.bqc

import ...

class AddSite : AppCompatActivity() {
    private lateinit var binding: ActivityAddSiteBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityAddSiteBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val documentId = intent.getStringExtra( name: "documentID")
        val name = binding.etSiteName
        val description = binding.etDescription
        val materialsUsed = binding.etMaterialsUsed
        val siteType = binding.toggleSiteType
        val currentWork = binding.etWorkInProgress
        val totalBudget = binding.etTotalBudget
        val spendings = binding.etSpent
        val startDate = binding.etStartDate
        val endDate = binding.etEndDate
        val siteStatus = binding.toggleSiteStatus
        val submitBtn = binding.btnAddSite
        val db = Firebase.firestore
        submitBtn.setOnClickListener {
            Log.v( tag: "Debug", msg: "Button Clicked")
            val siteData = hashMapOf(
                "name" to name.text.toString(),
                "description" to description.text.toString(),
                "materialsUsed" to materialsUsed.text.toString(),
                "siteType" to when(siteType.checkedButtonId){
                    R.id.btnResidential->"Residential"
                    R.id.btnCommercial->"Commercial"
                    R.id.btnIndustrial->"Industrial"
                    else->"Undefined"
                },
                "currentWork" to currentWork.text.toString(),
                "totalBudget" to totalBudget.text.toString(),
                "spendings" to spendings.text.toString(),
                "startDate" to startDate.text.toString(),
                "endDate" to endDate.text.toString(),
                "siteStatus" to when(siteStatus.checkedButtonId){
                    R.id.btntoStart->"To start"
```

```java
class AddSite : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        submitBtn.setOnClickListener {
                "siteStatus" to when(siteStatus.checkedButtonId){
                    R.id.btntoStart->"To start"
                    R.id.btnActive->"Active"
                    R.id.btnCompleted->"Completed"
                    else->"Undefined"
                },
                "qualityCheck" to "No data"
            )
            Log.v( tag: "Debug", msg: "Data is made going to add data")
            db.collection( collectionPath: "sites").add(siteData).addOnCompleteListener {
                Toast.makeText( context: this, text: "Site added successfully",Toast.LENGTH_SHORT).show()
                val intent = Intent( packageContext: this, MainActivity::class.java)
                startActivity(intent)
                finish()
            }.addOnFailureListener {
                Toast.makeText( context: this, text: "Failed to add document",Toast.LENGTH_SHORT).show()
                val intent = Intent( packageContext: this, MainActivity::class.java)
                startActivity(intent)
                finish()
            }
        }

        if (documentId!=null){
            submitBtn.text = "Update Info"
            db.collection( collectionPath: "sites").document( documentPath: documentId?:"").get().addOnSuccessListener {
                document->
                    if(document.exists()){
                        name.setText(document.getString( field: "name"))
                        description.setText(document.getString( field: "description"))
                        materialsUsed.setText(document.getString( field: "materialsUsed"))
                        currentWork.setText(document.getString( field: "currentWork"))
                        totalBudget.setText(document.getString( field: "totalBudget"))
                        spendings.setText(document.getString( field: "spendings"))
                        startDate.setText(document.getString( field: "startDate"))
                        endDate.setText(document.getString( field: "endDate"))
                        if(document.getString( field: "siteType")=="Residenial"){
                            siteType.check(R.id.btnResidential)
                        }
                        else if(document.getString( field: "siteType")=="Commercial") {
                            siteType.check(R.id.btnCommercial)
                        }
```

```
15      class AddSite : AppCompatActivity() {
17          override fun onCreate(savedInstanceState: Bundle?) {
77                  db.collection( collectionPath: "sites").document( documentPath: documentId?:"").get().addOnSuccessListener {
93                      }
94                      else if(document.getString( field: "siteType")=="Industrial"){
95                          siteType.check(R.id.btnIndustrial)
96                      }
97                      if(document.getString( field: "siteStatus")=="To start"){
98                          siteStatus.check(R.id.btntoStart)
99                      }
100                     else if(document.getString( field: "siteStatus")=="Active") {
101                         siteStatus.check(R.id.btnActive)
102                     }
103                     else if(document.getString( field: "siteStatus")=="Completed"){
104                         siteType.check(R.id.btnCompleted)
105                     }
106
107                 }
108                 }
109                 submitBtn.setOnClickListener {
110                     Log.v( tag: "Debug", msg: "Button Clicked")
111                     val siteData = hashMapOf(
112                         "name" to name.text.toString(),
113                         "description" to description.text.toString(),
114                         "materialsUsed" to materialsUsed.text.toString(),
115                         "siteType" to when(siteType.checkedButtonId){
116                             R.id.btnResidential->"Residential"
117                             R.id.btnCommercial->"Commercial"
118                             R.id.btnIndustrial->"Industrial"
119                             else->"Undefined"
120                         },
121                         "currentWork" to currentWork.text.toString(),
122                         "totalBudget" to totalBudget.text.toString(),
123                         "spendings" to spendings.text.toString(),
124                         "startDate" to startDate.text.toString(),
125                         "endDate" to endDate.text.toString(),
126                         "siteStatus" to when(siteStatus.checkedButtonId){
127                             R.id.btntoStart->"To start"
128                             R.id.btnActive->"Active"
129                             R.id.btnCompleted->"Completed"
130                             else->"Undefined"
131                         }
132                     )
133                     db.collection( collectionPath: "sites").document(documentId).update(siteData as Map<String,Any>).addOnCompleteListener {
```

```
133                     db.collection( collectionPath: "sites").document(documentId).update(siteData as Map<String,Any>).addOnCompleteListener {
134                         Toast.makeText( context: this,  text: "Site added successfully",Toast.LENGTH_SHORT).show()
135                         val intent = Intent( packageContext: this, MainActivity::class.java)
136                         startActivity(intent)
137                         finish()
138                     }.addOnFailureListener {
139                         Toast.makeText( context: this, text: "Failed to add document",Toast.LENGTH_SHORT).show()
140                         val intent = Intent( packageContext: this, MainActivity::class.java)
141                         startActivity(intent)
142                         finish()
143                     }
144                 }
145             }
146         }
147     }
```

Feedback

```kotlin
package com.example.bqc

import ...

class Feedback : AppCompatActivity() {
    private lateinit var binding: ActivityFeedbackBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityFeedbackBinding.inflate(layoutInflater)
        setContentView(binding.root)
        val db = Firebase.firestore
        binding.btnSubmitFeedback.setOnClickListener {
            val data = hashMapOf(
                "feedback" to binding.editFeedback.text.toString(),
                "rating" to binding.feedbackRating.rating.toInt()
            )
            db.collection( collectionPath: "feedback").add(data).addOnCompleteListener {
                Toast.makeText( context: this, text: "Feedback Sent.",Toast.LENGTH_SHORT).show()
                finish()
            }.addOnFailureListener {
                Toast.makeText( context: this, text: "Failed to send feedback.",Toast.LENGTH_SHORT).show()
                finish()
            }
        }

    }
}
```

Home Fragment

```kotlin
package com.example.bqc

import ...

class homefragment: Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view =  inflater.inflate(R.layout.homefragment,container, attachToRoot: false)
        val addSite = view.findViewById<CardView>(R.id.btnAddSite)
        addSite.setOnClickListener {
            val intent = Intent(requireContext(),AddSite::class.java)
            startActivity(intent)
        }
        val siteList = view.findViewById<CardView>(R.id.btnSiteList)
        siteList.setOnClickListener {
            val intent = Intent(requireContext(),SiteList::class.java)
            startActivity(intent)
        }
        val feedback = view.findViewById<CardView>(R.id.btnSubmitFeedback)
        feedback.setOnClickListener{
            val intent = Intent(requireContext(), Feedback::class.java)
            startActivity(intent)
        }
        val about = view.findViewById<CardView>(R.id.btnAbout)
        about.setOnClickListener{
            val intent = Intent(requireContext(), AboutScreen::class.java)
            startActivity(intent)
        }
        return view
    }
}
```

Material info fragment

```
1   package com.example.bqc
2
3   > import ...
11
12 </> class materialinfofragment: Fragment() {
13 @↑     override fun onCreateView(
14             inflater: LayoutInflater,
15             container: ViewGroup?,
16             savedInstanceState: Bundle?
17         ): View? {
18             val view = inflater.inflate(R.layout.materialinfo,container, attachToRoot: false)
19             val concrete = view.findViewById<Button>(R.id.btnConcrete)
20             val steel = view.findViewById<Button>(R.id.btnSteelBars)
21             val brick = view.findViewById<Button>(R.id.btnBricks)
22             val wiring = view.findViewById<Button>(R.id.btnWiring)
23             val waterproofing = view.findViewById<Button>(R.id.btnWaterproofing)
24             val plaster = view.findViewById<Button>(R.id.btnPlaster)
25             val fireSafety = view.findViewById<Button>(R.id.btnFireSafety)
26             val inflater = layoutInflater
27             concrete.setOnClickListener{
28                 val popupView = inflater.inflate(R.layout.concrete, root: null)
29                 val popupWindow = PopupWindow(
30                     popupView,
31                     ViewGroup.LayoutParams.MATCH_PARENT,
32                     ViewGroup.LayoutParams.WRAP_CONTENT,
33                     focusable: true
34                 )
35                 popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
36             }
37             steel.setOnClickListener{
38                 val popupView = inflater.inflate(R.layout.steel, root: null)
39                 val popupWindow = PopupWindow(
40                     popupView,
41                     ViewGroup.LayoutParams.MATCH_PARENT,
42                     ViewGroup.LayoutParams.WRAP_CONTENT,
43                     focusable: true
44                 )
45                 popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
46             }
47             brick.setOnClickListener{
48                 val popupView = inflater.inflate(R.layout.bricks, root: null)
49                 val popupWindow = PopupWindow(
50                     popupView,
51                     ViewGroup.LayoutParams.MATCH_PARENT,
```

```
50                     popupView,
51                     ViewGroup.LayoutParams.MATCH_PARENT,
52                     ViewGroup.LayoutParams.WRAP_CONTENT,
53                     focusable: true
54                 )
55                 popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
56             }
57             wiring.setOnClickListener{
58                 val popupView = inflater.inflate(R.layout.wiring, root: null)
59                 val popupWindow = PopupWindow(
60                     popupView,
61                     ViewGroup.LayoutParams.MATCH_PARENT,
62                     ViewGroup.LayoutParams.WRAP_CONTENT,
63                     focusable: true
64                 )
65                 popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
66             }
67             waterproofing.setOnClickListener{
68                 val popupView = inflater.inflate(R.layout.waterproofing, root: null)
69                 val popupWindow = PopupWindow(
70                     popupView,
71                     ViewGroup.LayoutParams.MATCH_PARENT,
72                     ViewGroup.LayoutParams.WRAP_CONTENT,
73                     focusable: true
74                 )
75                 popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
76             }
77             plaster.setOnClickListener{
78                 val popupView = inflater.inflate(R.layout.plaster, root: null)
79                 val popupWindow = PopupWindow(
80                     popupView,
81                     ViewGroup.LayoutParams.MATCH_PARENT,
82                     ViewGroup.LayoutParams.WRAP_CONTENT,
83                     focusable: true
84                 )
85                 popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
86             }
87             fireSafety.setOnClickListener{
88                 val popupView = inflater.inflate(R.layout.firesafety, root: null)
89                 val popupWindow = PopupWindow(
90                     popupView,
```

```
                ViewGroup.LayoutParams.WRAP_CONTENT,
                focusable: true
            )
            popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
        }
        plaster.setOnClickListener{
            val popupView = inflater.inflate(R.layout.plaster, root: null)
            val popupWindow = PopupWindow(
                popupView,
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                focusable: true
            )
            popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
        }
        fireSafety.setOnClickListener{
            val popupView = inflater.inflate(R.layout.firesafety, root: null)
            val popupWindow = PopupWindow(
                popupView,
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                focusable: true
            )
            popupWindow.showAtLocation(it, Gravity.CENTER, x: 0, y: 0)
        }
        return view
    }
}
```

Profile Fragment

```
class profilefragment: Fragment() {
    private lateinit var firebaseAuth: FirebaseAuth
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        firebaseAuth = FirebaseAuth.getInstance()
        val view = inflater.inflate(R.layout.profilefragment,container, attachToRoot: false)
        view.findViewById<TextView>(R.id.profile_email).text = firebaseAuth.currentUser?.email.toString()
        val profileName = view.findViewById<TextView>(R.id.profile_name)
        firebaseAuth.currentUser?.let {
            profileName.text = it.displayName.toString()
        }

        val signOut = view.findViewById<Button>(R.id.btn_sign_out)
        signOut.setOnClickListener {
            firebaseAuth.signOut()
            val intent = Intent(requireContext(), SignIn::class.java)
            startActivity(intent)
            requireActivity().finish()
        }

        val user = FirebaseAuth.getInstance().currentUser
        val deleteAccount = view.findViewById<Button>(R.id.btn_delete_account)
        deleteAccount.setOnClickListener {
            user?.delete()?.addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    // Account deleted successfully
                    val intent = Intent(requireContext(), SignUp::class.java)
                    intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
                    startActivity(intent)
                    requireActivity().finish()
                } else {
                    // Deletion failed (maybe the user needs to re-authenticate)
                    val error = task.exception?.message ?: "Account deletion failed."
                    Toast.makeText(requireContext(), error, Toast.LENGTH_LONG).show()
                }
            }
        }

        return view
    }
}
```

QualityCheck

```java
package com.example.bqc

import ...

class QualityQuiz : AppCompatActivity() {
    private lateinit var binding: ActivityQualityQuizBinding
    private var goodCount = 0
    private var averageCount = 0
    private var badCount = 0
    private val badMaterials = mutableListOf<String>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_quality_quiz)

        val resultTextView: TextView = findViewById(R.id.resultTextView)
        val checkButton: Button = findViewById(R.id.checkQualityButton)
        val db = Firebase.firestore
        val documentID = intent.getStringExtra( name: "documentID")
        checkButton.setOnClickListener {
            // Reset all counters
            goodCount = 0
            averageCount = 0
            badCount = 0
            badMaterials.clear()

            // Run evaluations
            evaluate(
                findViewById(R.id.brickClassGroup),
                mapOf(
                    R.id.brickOptionA to "Bad",
                    R.id.brickOptionB to "Average",
                    R.id.brickOptionC to "Good",
                    R.id.brickOptionD to "Good"
                ),
                materialName: "Bricks"
            )

            evaluate(
                findViewById(R.id.steelGradeGroup),
                mapOf(
                    R.id.steelOptionA to "Average",
                    R.id.steelOptionB to "Good",
                    R.id.steelOptionC to "Good",
```

```java
            evaluate(
                findViewById(R.id.steelGradeGroup),
                mapOf(
                    R.id.steelOptionA to "Average",
                    R.id.steelOptionB to "Good",
                    R.id.steelOptionC to "Good",
                    R.id.steelOptionD to "Bad"
                ),
                materialName: "TMT Steel"
            )

            evaluate(
                findViewById(R.id.concreteGradeGroup),
                mapOf(
                    R.id.concreteOptionA to "Bad",
                    R.id.concreteOptionB to "Average",
                    R.id.concreteOptionC to "Good",
                    R.id.concreteOptionD to "Good"
                ),
                materialName: "Concrete"
            )

            evaluate(
                findViewById(R.id.waterAbsorptionGroup),
                mapOf(
                    R.id.absorpOptionA to "Bad",
                    R.id.absorpOptionB to "Average",
                    R.id.absorpOptionC to "Good",
                    R.id.absorpOptionD to "Good"
                ),
                materialName: "Brick Water Absorption"
            )

            evaluate(
                findViewById(R.id.waterproofingGroup),
                mapOf(
                    R.id.waterproofOptionA to "Bad",
                    R.id.waterproofOptionB to "Average",
                    R.id.waterproofOptionC to "Good",
                    R.id.waterproofOptionD to "Good"
                ),
```

```
                materialName: "Waterproofing"
            )

            evaluate(
                findViewById(R.id.wiringGroup),
                mapOf(
                    R.id.wiringOptionA to "Bad",
                    R.id.wiringOptionB to "Average",
                    R.id.wiringOptionC to "Good",
                    R.id.wiringOptionD to "Good"
                ),
                materialName: "Electrical Wiring"
            )

            evaluate(
                findViewById(R.id.fireRatingGroup),
                mapOf(
                    R.id.fireOptionA to "Bad",
                    R.id.fireOptionB to "Average",
                    R.id.fireOptionC to "Average",
                    R.id.fireOptionD to "Good"
                ),
                materialName: "Fire Door Rating"
            )

            evaluate(
                findViewById(R.id.plasterThicknessGroup),
                mapOf(
                    R.id.plasterOptionA to "Bad",
                    R.id.plasterOptionB to "Average",
                    R.id.plasterOptionC to "Good",
                    R.id.plasterOptionD to "Good"
                ),
                materialName: "Plaster Thickness"
            )

            val result = buildResultReport()
            resultTextView.text = result
            Log.v( tag: "debugApp", msg: "Changed result, now updating data")
            if(documentID!=null){
            db.collection( collectionPath: "sites").document( documentPath: documentID?:"").update( field: "qualityCheck",result).addOnCompleteListener {
```

```
class QualityQuiz : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        checkButton.setOnClickListener {
            if(documentID!=null){
            db.collection( collectionPath: "sites").document( documentPath: documentID?:"").update( field: "qualityCheck",result).addOnCompleteListener {
                Toast.makeText( context: this, text: "Updated quality check in the database",Toast.LENGTH_SHORT).show()
            }.addOnFailureListener {
                Toast.makeText( context: this, text: "Failed to update quality check in the database",Toast.LENGTH_SHORT).show()
            }
            }
            else{
                Toast.makeText( context: this, text: "Invalid documentID",Toast.LENGTH_SHORT).show()
            }
        }
    }

    private fun evaluate(group: RadioGroup, answerKey: Map<Int, String>, materialName: String) {
        val selectedId = group.checkedRadioButtonId
        when (answerKey[selectedId]) {
            "Good" -> goodCount++
            "Average" -> averageCount++
            "Bad" -> {
                badCount++
                badMaterials.add(materialName)
            }
        }
    }

    private fun buildResultReport(): String {
        val total = goodCount + averageCount + badCount

        val summary = when {
            goodCount >= 6 -> "✅ Most materials are GOOD. Construction quality looks solid!"
            averageCount >= 4 -> "⚠️ Several materials are AVERAGE. May need a few improvements."
            badCount >= 2 -> "❌ Warning: Multiple BAD materials detected. Please take action!"
            else -> "✅ Mostly acceptable, with a few areas to monitor."
        }

        val badList = if (badMaterials.isNotEmpty()) {
            "\n\n🔴 Bad Materials Used:\n• " + badMaterials.joinToString( separator: "\n• ")
        } else {
            "\n\n🎉 No bad materials detected!"
        }
```

```java
            val badList = if (badMaterials.isNotEmpty()) {
                "\n\n🔴 Bad Materials Used:\n• " + badMaterials.joinToString( separator: "\n• ")
            } else {
                "\n\n🎉 No bad materials detected!"
            }

            return """
                📋 Quality Check Report:
                • Good: $goodCount
                • Average: $averageCount
                • Bad: $badCount
                • Total Checks: $total

                📊 Verdict:
                $summary

                $badList
            """.trimIndent()
        }
    }
```

Report Issues

```java
package com.example.bqc

import ...

class ReportIssues : AppCompatActivity() {
    private lateinit var binding: ActivityReportIssuesBinding
    private lateinit var firebaseAuth : FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding=ActivityReportIssuesBinding.inflate(layoutInflater)
        setContentView(binding.root)
        val db = Firebase.firestore
        val firebaseAuth = FirebaseAuth.getInstance()
        val documentId = intent.getStringExtra( name: "documentID")
        val categoryDropdown = binding.dropdownCategory
        val descriptionInput = binding.inputDescription


        val categories = arrayOf("App Crash", "Bug Report", "Payment Issue", "Other")
        val adapter = ArrayAdapter( context: this, android.R.layout.simple_list_item_1, categories)
        categoryDropdown.setAdapter(adapter)
        binding.btnSubmit.setOnClickListener {
            val category = categoryDropdown.text.toString().trim()
            val description = descriptionInput.text.toString().trim()
            if (category.isEmpty() || description.isEmpty()) {
                Toast.makeText( context: this, text: "Please fill in all fields", Toast.LENGTH_SHORT).show()
            } else {
                db.collection( collectionPath: "sites").document( documentPath: documentId?:"").update( field: "report", value: "${firebaseAuth.currentUser?.email}'s report
                    Toast.makeText( context: this, text: "Issue reported successfully!", Toast.LENGTH_LONG).show()
                    finish()
                }.addOnFailureListener {
                    Toast.makeText( context: this, text: "Failed to submit report",Toast.LENGTH_SHORT).show()
                    finish()
                }
            }
        }
    }
}
```

Site Actions

```java
package com.example.bqc

import ...

class SiteActions : AppCompatActivity() {
    private lateinit var binding: ActivitySiteActionsBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivitySiteActionsBinding.inflate(layoutInflater)
        setContentView(binding.root)
        val documentID = intent.getStringExtra( name: "documentID")
        binding.qualityCheck.setOnClickListener {
            val i =Intent( packageContext: this,MainActivity::class.java)
            i.putExtra( name: "documentID",documentID)
        }
        binding.qualityCheck.setOnClickListener {
            val i = Intent( packageContext: this,QualityQuiz::class.java)
            i.putExtra( name: "documentID",documentID)
            startActivity(i)
        }
        binding.report.setOnClickListener {
            val i =Intent( packageContext: this,ReportIssues::class.java)
            i.putExtra( name: "documentID",documentID)
            startActivity(i)
        }
    }
}
```

Site Details

```
package com.example.bqc

import ...

class SiteDetails : AppCompatActivity() {
    private lateinit var binding: ActivitySiteDetailsBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivitySiteDetailsBinding.inflate(layoutInflater)
        setContentView(binding.root)
        val db = Firebase.firestore
        val documentID = intent.getStringExtra( name: "documentID")
        if(documentID==null){
            Toast.makeText( context: this,  text: "null documentID",Toast.LENGTH_SHORT).show()
            finish()
        }
        db.collection( collectionPath: "sites").document( documentPath: documentID?:"").get().addOnSuccessListener {
            document->
            if(document.exists()){
                binding.name.text = document.getString( field: "name")
                binding.description.text=document.getString( field: "description")
                binding.materialsUsed.text="Materials used: "+document.getString( field: "materialsUsed")
                binding.siteTypeText.text="Site Type: "+ document.getString( field: "siteType")
                binding.currentWork.text="Current Work: "+document.getString( field: "currentWork")
                binding.totalBudget.text="Total Budget: "+document.getString( field: "totalBudget")
                binding.spendings.text="Spendings: "+document.getString( field: "spendings")
                binding.startDate.text="Start Date: "+document.getString( field: "startDate")
                binding.endDate.text="End Date: "+document.getString( field: "endDate")
                binding.siteStatusText.text="Site Status: "+document.getString( field: "siteStatus")
                binding.qualityDetails.text = document.getString( field: "qualityCheck")
                if(document.getString( field: "siteStatus")=="Active"){
                    binding.siteStatusText.setTextColor(Color.parseColor( colorString: "#00FF00"))
                }else if(document.getString( field: "siteStatus")=="Completed"){
                    binding.siteStatusText.setTextColor(Color.parseColor( colorString: "#0000FF"))
                }else if(document.getString( field: "siteStatus")=="To Start"){
                    binding.siteStatusText.setTextColor(Color.parseColor( colorString: "#FFFF00"))
                }
            }
        }.addOnFailureListener {
            Toast.makeText( context: this, text: "Failed to fetch data",Toast.LENGTH_SHORT).show()
            finish()
        }
        binding.nextBtn.setOnClickListener {
```

```
            }
        }.addOnFailureListener {
            Toast.makeText( context: this, text: "Failed to fetch data",Toast.LENGTH_SHORT).show()
            finish()
        }
        binding.nextBtn.setOnClickListener {
            val i = Intent( packageContext: this, SiteActions::class.java)
            i.putExtra( name: "documentID",documentID)
            startActivity(i)
        }
        binding.updateBtn.setOnClickListener {
            val i = Intent( packageContext: this, AddSite::class.java)
            i.putExtra( name: "documentID",documentID)
            startActivity(i)
        }
    }
}
```

Site List

```java
package com.example.bqc

import ...

data class ConstructionSite(
    val name: String,
    val status: String,
    val address: String,
    val documentID: String
)

class SiteAdapter(private val siteList: List<ConstructionSite>) :
    RecyclerView.Adapter<SiteAdapter.SiteViewHolder>() {

    class SiteViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val siteName: TextView = itemView.findViewById(R.id.textViewSiteName)
        val siteStatus: TextView = itemView.findViewById(R.id.textViewStatus)
        val siteAddress: TextView =  itemView.findViewById(R.id.textViewAddress)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): SiteViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_site_card, parent, attachToRoot: false)
        return SiteViewHolder(view)
    }

    override fun onBindViewHolder(holder: SiteViewHolder, position: Int) {
        val site = siteList[position]
        holder.siteName.text = site.name
        holder.siteStatus.text = "Status: ${site.status}"
        holder.siteAddress.text = site.address
        if(site.status.toString()=="Active"){
            holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_active_background)
        }
        if(site.status.toString()=="Completed"){
            holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_completed_background)
        }
        if(site.status.toString()=="To start"){
            holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_ongoing_background)
        }

        holder.itemView.setOnClickListener{
            val context = holder.itemView.context
            val intent = Intent(holder.itemView.context,SiteDetails::class.java)
```

```java
class SiteAdapter(private val siteList: List<ConstructionSite>) :
    override fun onBindViewHolder(holder: SiteViewHolder, position: Int) {
        if(site.status.toString()=="To start"){
            holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_ongoing_background)
        }

        holder.itemView.setOnClickListener{
            val context = holder.itemView.context
            val intent = Intent(holder.itemView.context,SiteDetails::class.java)
            intent.putExtra( name: "documentID", site.documentID)
            context.startActivity(intent)
        }
    }

    override fun getItemCount(): Int = siteList.size
}


class SiteList : AppCompatActivity() {
    private lateinit var recyclerView: RecyclerView
    private lateinit var adapter: SiteAdapter
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_site_list)
        val siteList = mutableListOf<ConstructionSite>()
        val db = Firebase.firestore
        setContentView(R.layout.activity_site_list)
        recyclerView = findViewById<RecyclerView>(R.id.recyclerView)
        recyclerView.layoutManager = LinearLayoutManager( context: this)
        adapter = SiteAdapter(siteList)
        recyclerView.adapter = adapter
        val loadingWheel = findViewById<ProgressBar>(R.id.loadingWheel)
        db.collection( collectionPath: "sites").get().addOnSuccessListener {
                querySnapshot->
            for (document in querySnapshot){
                val documentId = document.id
                siteList.add(ConstructionSite( name: document.getString( field: "name")?:"Unknown", status: document.getString( field: "siteStatus")?:"Unknown", address: do
            }
            loadingWheel.visibility = View.INVISIBLE
            adapter.notifyDataSetChanged()
        }.addOnFailureListener {
            Toast.makeText( context: this, text: "Failed to fetch data", Toast.LENGTH_SHORT).show()
        }
```