



**L** OVELY  
**P** ROFESSIONAL  
**U** NIVERSITY

**PROJECT NAME – BuildQcheck**

**Group2**

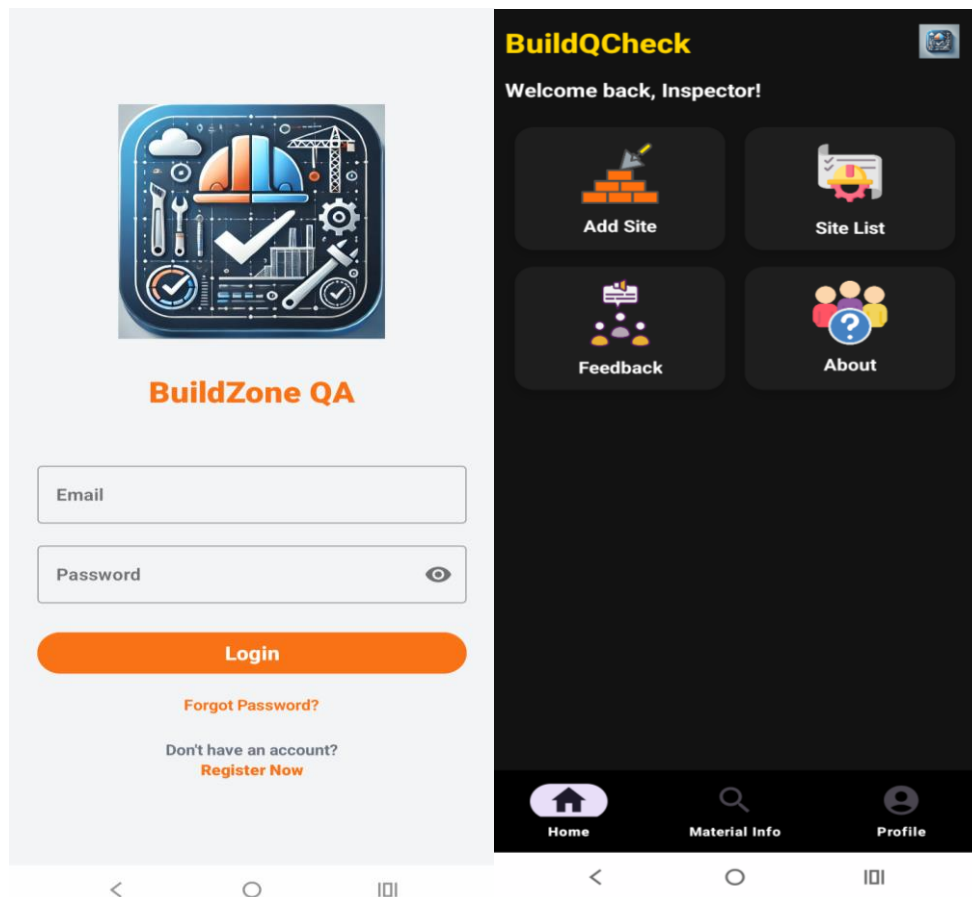
<b>S.No</b>	<b>Student Name</b>	<b>Reg Number</b>	<b>Roll No</b>
<b>1</b>	Aditya Malik	12210018	34
<b>2</b>	Uma Maheswari	12217139	09
<b>3</b>	Uddipan Biswas	12210388	06

## 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

Aditya Malik  
aditya14442@gmail.com


Sign Out

Delete Account

Home

Material Info

Profile

New Construction Site

1. Basic Info

Site Name

Description

Materials Used

RESIDEN ...

COMMER ...

INDUSTR ...

2. Work in Progress

Describe Current Work

3. Budget Information

Total Budget

Spent

4. Project Timeline

Start Date

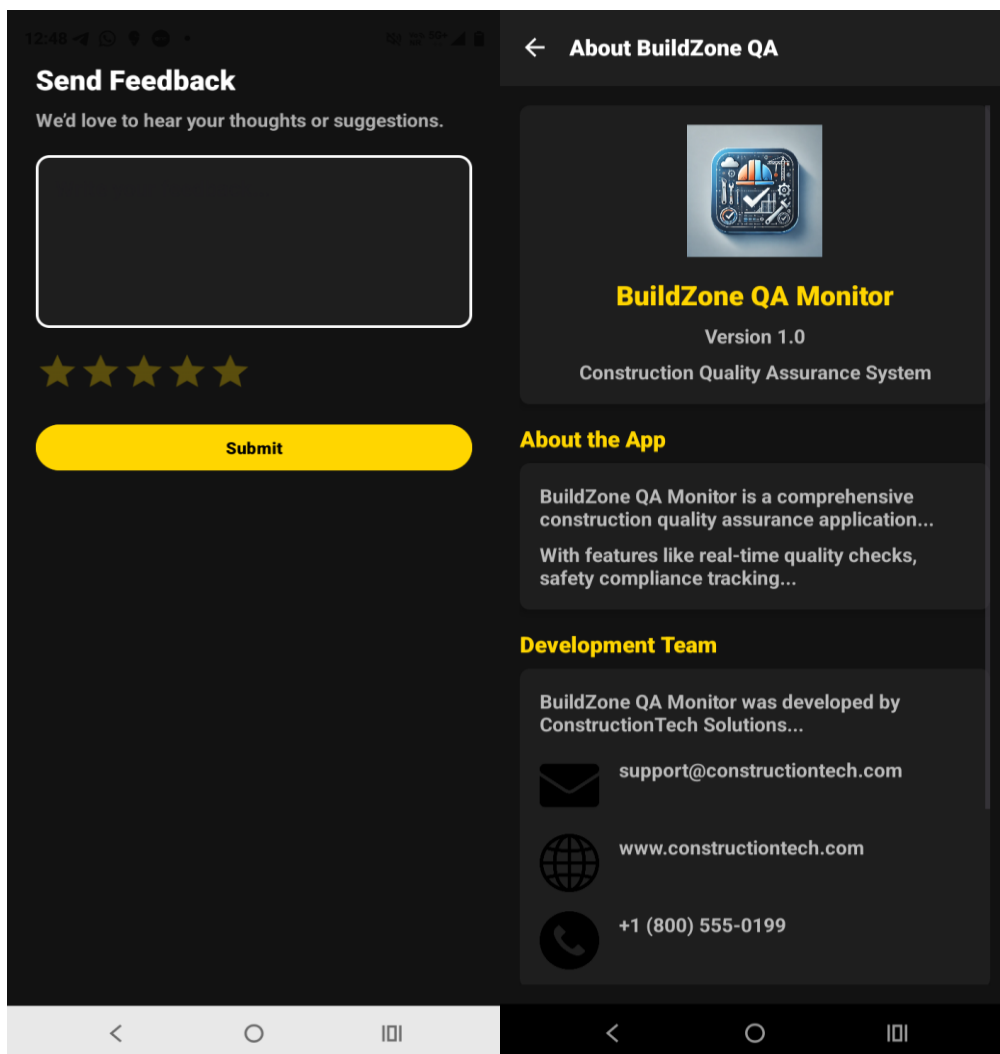
End Date

TO START

ACTIVE

COMPLE...

## Feedback and About App



Materials Info

Reference Standards

Concrete

Steel Bars

Bricks

Wiring

Waterproofing

Plaster

Fire Safety

IS 456:2000 – Plain Reinforced Concrete

Concrete Grades

• M15, M20, M25, M30, etc.

• Minimum: M20 for RCC; M10/M15 for PCC

Mix Design

• Proportioning of cement, sand, aggregate, water

• Refer: IS 10262

Durability Requirements

• Min. cement content

• Max. water-cement ratio

• Cover as per exposure (mild/moderate/severe)

Workmanship Curing

• Best practices for mixing, placing, compacting

• Curing: 7 days (ordinary cement), 10 days (blended cement)

Reinforcement Detailing

• Bar spacing, anchorage, lap length, cover

• Use TMT bars per IS 1786

Structural Design

• Limit State Method (preferred)

• Working Stress Method (deprecated)

Home

Material Info

Profile

Home

Material Info

Profile



## 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

### 1. Basic Info

Site Name  
Test

Description  
test

Materials Used  
test

RESIDEN...

COMMER...

INDUSTR...

### 2. Work in Progress

Describe Current Work  
testing app

### 3. Budget Information

Total Budget  
10000

Spent  
5000

### 4. Project Timeline

Start Date  
10

End Date  
20

TO START

ACTIVE

COMPLE...

Update Info

Test

test

Materials used: test

Site Type: Industrial

Current Work: testing app

Total Budget: 10000

Spending: 5000

Start Date: 10

End Date: 20

📄 Quality Check Report:

- Good: 3
- Average: 5
- Bad: 0
- Total Checks: 8

📄 Verdict:

⚠️ Several materials are AVERAGE. May need a few improvements.

🎉 No bad materials detected!

Site Status: To start

Next

Update data



## SIGN UP

```
1 package com.example.bqc
2
3 > import
4
13
14 class SignUp : AppCompatActivity() {
15     private lateinit var binding: ActivitySignUpBinding
16     private lateinit var firebaseAuth: FirebaseAuth
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         enableEdgeToEdge()
20
21         firebaseAuth = FirebaseAuth.getInstance()
22
23         binding = ActivitySignUpBinding.inflate(layoutInflater)
24         setContentView(binding.root)
25
26         binding.btnSignup.setOnClickListener{
27             val name = binding.etFullname.text.toString()
28             val email = binding.etEmail.text.toString()
29             val password = binding.etPassword.text.toString()
30             val confPass = binding.etConfirmPassword.text.toString()
31
32             if (name.isNotEmpty() && email.isNotEmpty() && password.isNotEmpty() && confPass.isNotEmpty()){
33                 if (password == confPass){
34                     firebaseAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener{
35                         if(it.isSuccessful){
36                             val profileUpdates = UserProfileChangeRequest {
37                                 displayName = name
38                             }
39                             firebaseAuth.currentUser?.updateProfile(profileUpdates)
40                             val user = firebaseAuth.currentUser
41                             user?.sendEmailVerification()?.addOnCompleteListener{
42                                 firebaseAuth.signOut()
43                                 if(it.isSuccessful){
44                                     Toast.makeText(context, this, text: "Check your mail for verification link.", Toast.LENGTH_LONG).show()
45                                 }
46                                 else{
47                                     Toast.makeText(context, this, it.exception.toString(), Toast.LENGTH_SHORT).show()
48                                 }
49                             }
50                         }
51                     }
52                 }
53                 else{
54                     Toast.makeText(context, this, text: "${it.exception.toString()}", Toast.LENGTH_LONG).show()
55                 }
56             }
57         }
58     }
59 }
```

```

50         }
51         else{
52             Toast.makeText(context: this, text: "${it.exception.toString()}", Toast.LENGTH_LONG).show()
53         }
54     }
55 }
56 else{
57     Toast.makeText(context: this, text: "Passwords do not match.", Toast.LENGTH_SHORT).show()
58 }
59 }
60 else{
61     Toast.makeText(context: this, text: "One or more field is empty.", Toast.LENGTH_SHORT).show()
62 }
63 }
64
65
66
67
68 binding.tvBackToSignIn.setOnClickListener{
69     val i = Intent(context: this, SignIn::class.java)
70     startActivity(i)
71 }
72
73 }
74
75 }

```

## Sign in

```

1 package com.example.bgc
2
3 > import ...
4
5 class SignIn : AppCompatActivity() {
6     private lateinit var binding: ActivitySignInBinding
7     private lateinit var firebaseAuth: FirebaseAuth
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10         enableEdgeToEdge()
11
12         firebaseAuth = FirebaseAuth.getInstance()
13
14         binding = ActivitySignInBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16
17         binding.tvForgotPassword.setOnClickListener{
18             val i = Intent(context: this, ForgotPassword::class.java)
19             startActivity(i)
20         }
21         binding.btnSignIn.setOnClickListener{
22             val email = binding.etEmail.text.toString()
23             val password = binding.etPassword.text.toString()
24             if(email.isNotEmpty() && password.isNotEmpty()){
25                 firebaseAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener{
26                     if(it.isSuccessful){
27                         val user = firebaseAuth.currentUser
28                         if(user != null) {
29                             if (user.isEmailVerified) {
30                                 val i = Intent(context: this, MainActivity::class.java)
31                                 startActivity(i)
32                             } else {
33                                 Toast.makeText(
34                                     context: this,
35                                     text: "Your email is not verified. Check your mail for verification link",
36                                     Toast.LENGTH_LONG
37                                 ).show()
38                             }
39                         }
40                     }
41                     else{
42                         Toast.makeText(context: this, it.exception.toString(), Toast.LENGTH_SHORT).show()
43                     }
44                 }
45             }
46         }
47     }
48 }
49
50 }
51
52 }

```

```

51         }
52     }
53 }
54 else{
55     Toast.makeText( context: this, text: "One or more fields is empty.", Toast.LENGTH_SHORT).show()
56 }
57 }
58
59
60 binding.tvSignup.setOnClickListener{
61     val i = Intent( packageContext: this,SignUp::class.java)
62     startActivity(i)
63 }
64 }
65
66 override fun onStart() {
67     super.onStart()
68     firebaseAuth = FirebaseAuth.getInstance()
69     val user = firebaseAuth.currentUser
70     if(user!=null && user.isEmailVerified){
71         val i = Intent( packageContext: this,MainActivity::class.java)
72         startActivity(i)
73     }
74 }

```

## Forgot Password

```

1 package com.example.bqc
2
3 > import ...
4
12
13 class ForgotPassword : AppCompatActivity() {
14     private lateinit var firebaseAuth : FirebaseAuth
15     private lateinit var binding : ActivityForgotPasswordBinding
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18         enableEdgeToEdge()
19         binding = ActivityForgotPasswordBinding.inflate(layoutInflater)
20         setContentView(binding.root)
21         firebaseAuth = FirebaseAuth.getInstance()
22         binding.btnResetpassword.setOnClickListener {
23             val email = binding.etEmail.text.toString().trim()
24             if (email.isNotEmpty()){
25                 firebaseAuth.sendPasswordResetEmail(email).addOnCompleteListener(this){task->
26                     if (task.isSuccessful){
27                         Toast.makeText( context: this, text: "Check your mail for password reset link.", Toast.LENGTH_LONG).show()
28                         val i = Intent( packageContext: this,SignIn::class.java)
29                         startActivity(i)
30                         finish()
31                     }
32                     else{
33                         Toast.makeText( context: this,task.exception.toString(), Toast.LENGTH_SHORT).show()
34                     }
35                 }
36             }
37             else{
38                 Toast.makeText( context: this, text: "Email cannot be empty", Toast.LENGTH_SHORT).show()
39             }
40         }
41     }
42 }
43 }

```

## Splash Screen

```
1 package com.example.bqc
2
3 > import ...
4
5
6
7
8
9
10 class SplashScreen : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         enableEdgeToEdge()
14         setContentView(R.layout.activity_splash_screen)
15         android.os.Handler().postDelayed({
16             val i = Intent(packageContext, SignIn::class.java)
17             startActivity(i)
18             finish()
19         }, delayMillis: 1000)
20     }
21 }
```

## About Screen

```
1 package com.example.bqc
2
3 > import ...
4
5
6
7
8
9
10 class AboutScreen : AppCompatActivity() {
11     private lateinit var binding: ActivityAboutScreenBinding
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         binding = ActivityAboutScreenBinding.inflate(layoutInflater)
16         setContentView(binding.root)
17
18         setSupportActionBar(binding.toolbar)
19         supportActionBar?.setDisplayHomeAsUpEnabled(true)
20         supportActionBar?.title = ""
21         val versionName = packageManager.getPackageInfo(packageName, 0).versionName
22         binding.txtVersion.text = "Version $versionName"
23     }
24
25     fun openPrivacyPolicy(view: View) {
26         val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.constructiontech.com/privacy"))
27         startActivity(intent)
28     }
29
30     fun openTermsOfService(view: View) {
31         val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.constructiontech.com/terms"))
32         startActivity(intent)
33     }
34
35     override fun onSupportNavigateUp(): Boolean {
36         finish()
37         return true
38     }
39 }
40
41
42 }
```

## Add Site

```

1 package com.example.bqc
2
3 > import androidx.appcompat.app.AppCompatActivity
4
14 class AddSite : AppCompatActivity() {
15     private lateinit var binding: ActivityAddSiteBinding
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18         enableEdgeToEdge()
19         binding = ActivityAddSiteBinding.inflate(layoutInflater)
20         setContentView(binding.root)
21
22         val documentId = intent.getStringExtra("documentID")
23         val name = binding.etSiteName
24         val description = binding.etDescription
25         val materialsUsed = binding.etMaterialsUsed
26         val siteType = binding.toggleSiteType
27         val currentWork = binding.etWorkInProgress
28         val totalBudget = binding.etTotalBudget
29         val spendings = binding.etSpent
30         val startDate = binding.etStartDate
31         val endDate = binding.etEndDate
32         val siteStatus = binding.toggleSiteStatus
33         val submitBtn = binding.btnAddSite
34         val db = Firebase.firestore
35         submitBtn.setOnClickListener {
36             Log.v(tag="Debug", msg="Button Clicked")
37             val siteData = hashMapOf(
38                 "name" to name.text.toString(),
39                 "description" to description.text.toString(),
40                 "materialsUsed" to materialsUsed.text.toString(),
41                 "siteType" to when(siteType.checkedButtonId){
42                     R.id.btnResidential->"Residential"
43                     R.id.btnCommercial->"Commercial"
44                     R.id.btnIndustrial->"Industrial"
45                     else->"Undefined"
46                 },
47                 "currentWork" to currentWork.text.toString(),
48                 "totalBudget" to totalBudget.text.toString(),
49                 "spendings" to spendings.text.toString(),
50                 "startDate" to startDate.text.toString(),
51                 "endDate" to endDate.text.toString(),
52                 "siteStatus" to when(siteStatus.checkedButtonId){
53                     R.id.btnToStart->"To start"

```

```

54                     R.id.btnActive->"Active"
55                     R.id.btnCompleted->"Completed"
56                     else->"Undefined"
57                 },
58                 "qualityCheck" to "No data"
59             )
60             Log.v(tag="Debug", msg="Data is made going to add data")
61             db.collection(collectionPath="sites").add(siteData).addOnCompleteListener {
62                 Toast.makeText(context=this, text="Site added successfully", Toast.LENGTH_SHORT).show()
63                 val intent = Intent(packageContext=this, MainActivity::class.java)
64                 startActivity(intent)
65                 finish()
66             }.addOnFailureListener {
67                 Toast.makeText(context=this, text="Failed to add document", Toast.LENGTH_SHORT).show()
68                 val intent = Intent(packageContext=this, MainActivity::class.java)
69                 startActivity(intent)
70                 finish()
71             }
72         }
73     }
74
75     if (documentId!=null){
76         submitBtn.text = "Update Info"
77         db.collection(collectionPath="sites").document(documentPath=documentId?.toString()).get().addOnSuccessListener {
78             document->
79             if(document.exists()){
80                 name.setText(document.getString(field="name"))
81                 description.setText(document.getString(field="description"))
82                 materialsUsed.setText(document.getString(field="materialsUsed"))
83                 currentWork.setText(document.getString(field="currentWork"))
84                 totalBudget.setText(document.getString(field="totalBudget"))
85                 spendings.setText(document.getString(field="spendings"))
86                 startDate.setText(document.getString(field="startDate"))
87                 endDate.setText(document.getString(field="endDate"))
88                 if(document.getString(field="siteType")==="Residential"){
89                     siteType.check(R.id.btnResidential)
90                 }
91                 else if(document.getString(field="siteType")==="Commercial"){
92                     siteType.check(R.id.btnCommercial)
93                 }

```

```

15 class AddSite : AppCompatActivity() {
16     override fun onCreate(savedInstanceState: Bundle?) {
17         db.collection(collectionPath: "sites").document(documentId: "").get().addOnSuccessListener {
18             }
19             else if(document.getString(field: "siteType")=="Industrial"){
20                 siteType.check(R.id.btnIndustrial)
21             }
22             if(document.getString(field: "siteStatus")=="To start"){
23                 siteStatus.check(R.id.btnToStart)
24             }
25             else if(document.getString(field: "siteStatus")=="Active") {
26                 siteStatus.check(R.id.btnActive)
27             }
28             else if(document.getString(field: "siteStatus")=="Completed"){
29                 siteType.check(R.id.btnCompleted)
30             }
31         }
32     }
33     submitBtn.setOnClickListener {
34         Log.v(tag: "Debug", msg: "Button Clicked")
35         val siteData = hashMapOf(
36             "name" to name.text.toString(),
37             "description" to description.text.toString(),
38             "materialsUsed" to materialsUsed.text.toString(),
39             "siteType" to when(siteType.checkedButtonId){
40                 R.id.btnResidential->"Residential"
41                 R.id.btnCommercial->"Commercial"
42                 R.id.btnIndustrial->"Industrial"
43                 else->"Undefined"
44             },
45             "currentWork" to currentWork.text.toString(),
46             "totalBudget" to totalBudget.text.toString(),
47             "spendings" to spendings.text.toString(),
48             "startDate" to startDate.text.toString(),
49             "endDate" to endDate.text.toString(),
50             "siteStatus" to when(siteStatus.checkedButtonId){
51                 R.id.btnToStart->"To start"
52                 R.id.btnActive->"Active"
53                 R.id.btnCompleted->"Completed"
54                 else->"Undefined"
55             }
56         )
57         db.collection(collectionPath: "sites").document(documentId).update(siteData as Map<String,Any>).addOnCompleteListener {
58             Toast.makeText(context: this, text: "Site added successfully",Toast.LENGTH_SHORT).show()
59             val intent = Intent(packageContext: this, MainActivity::class.java)
60             startActivity(intent)
61             finish()
62             .addOnFailureListener {
63                 Toast.makeText(context: this, text: "Failed to add document",Toast.LENGTH_SHORT).show()
64                 val intent = Intent(packageContext: this, MainActivity::class.java)
65                 startActivity(intent)
66                 finish()
67             }
68         }
69     }
70 }
71 }
72 }

```

Feedback

```

1 package com.example.bqc
2
3 > import ...
12
13 class Feedback : AppCompatActivity() {
14     private lateinit var binding: ActivityFeedbackBinding
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         enableEdgeToEdge()
18         binding = ActivityFeedbackBinding.inflate(layoutInflater)
19         setContentView(binding.root)
20         val db = Firebase.firestore
21         binding.btnSubmitFeedback.setOnClickListener {
22             val data = hashMapOf(
23                 "feedback" to binding.editFeedback.text.toString(),
24                 "rating" to binding.feedbackRating.rating.toInt()
25             )
26             db.collection(collectionPath: "feedback").add(data).addOnCompleteListener {
27                 Toast.makeText(context: this, text: "Feedback Sent.", Toast.LENGTH_SHORT).show()
28                 finish()
29             }.addOnFailureListener {
30                 Toast.makeText(context: this, text: "Failed to send feedback.", Toast.LENGTH_SHORT).show()
31                 finish()
32             }
33         }
34     }
35 }
36

```

## Home Fragment

```

1 package com.example.bqc
2
3 > import ...
12
13 class homefragment: Fragment() {
14     override fun onCreateView(
15         inflater: LayoutInflater,
16         container: ViewGroup?,
17         savedInstanceState: Bundle?
18     ): View? {
19         val view = inflater.inflate(R.layout.homefragment, container, attachToRoot: false)
20         val addSite = view.findViewById<CardView>(R.id.btnAddSite)
21         addSite.setOnClickListener {
22             val intent = Intent(requireContext(), AddSite::class.java)
23             startActivity(intent)
24         }
25         val siteList = view.findViewById<CardView>(R.id.btnSiteList)
26         siteList.setOnClickListener {
27             val intent = Intent(requireContext(), SiteList::class.java)
28             startActivity(intent)
29         }
30         val feedback = view.findViewById<CardView>(R.id.btnSubmitFeedback)
31         feedback.setOnClickListener {
32             val intent = Intent(requireContext(), Feedback::class.java)
33             startActivity(intent)
34         }
35         val about = view.findViewById<CardView>(R.id.btnAbout)
36         about.setOnClickListener {
37             val intent = Intent(requireContext(), AboutScreen::class.java)
38             startActivity(intent)
39         }
40         return view
41     }
42 }

```

## Material info fragment

```
1 package com.example.bqc
2
3 > import ...
4
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,
```



```

72         ViewGroup.LayoutParams.WRAP_CONTENT,
73         focusable: true
74     )
75     popupWindow.showAtLocation(it, Gravity.CENTER, 0, 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, 0, 0)
86 }
87 fireSafety.setOnClickListener{
88     val popupView = inflater.inflate(R.layout.firesafety, root: null)
89     val popupWindow = PopupWindow(
90         popupView,
91         ViewGroup.LayoutParams.MATCH_PARENT,
92         ViewGroup.LayoutParams.WRAP_CONTENT,
93         focusable: true
94     )
95     popupWindow.showAtLocation(it, Gravity.CENTER, 0, 0)
96 }
97 return view
98 }
99 }

```

## Profile Fragment

```

15 class profilefragment: Fragment() {
16     private lateinit var firebaseAuth: FirebaseAuth
17     override fun onCreateView(
18         inflater: LayoutInflater,
19         container: ViewGroup?,
20         savedInstanceState: Bundle?
21     ): View? {
22         firebaseAuth = FirebaseAuth.getInstance()
23         val view = inflater.inflate(R.layout.profilefragment, container, attachToRoot: false)
24         view.findViewById<TextView>(R.id.profile_email).text = firebaseAuth.currentUser?.email.toString()
25         val profileName = view.findViewById<TextView>(R.id.profile_name)
26         firebaseAuth.currentUser?.let {
27             profileName.text = it.displayName.toString()
28         }
29
30         val signOut = view.findViewById<Button>(R.id.btn_sign_out)
31         signOut.setOnClickListener {
32             firebaseAuth.signOut()
33             val intent = Intent(requireContext(), SignIn::class.java)
34             startActivity(intent)
35             requireActivity().finish()
36         }
37
38         val user = FirebaseAuth.getInstance().currentUser
39         val deleteAccount = view.findViewById<Button>(R.id.btn_delete_account)
40         deleteAccount.setOnClickListener {
41             user?.delete()?.addOnCompleteListener { task ->
42                 if (task.isSuccessful) {
43                     // Account deleted successfully
44                     val intent = Intent(requireContext(), SignUp::class.java)
45                     intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
46                     startActivity(intent)
47                     requireActivity().finish()
48                 } else {
49                     // Deletion failed (maybe the user needs to re-authenticate)
50                     val error = task.exception?.message ?: "Account deletion failed."
51                     Toast.makeText(requireContext(), error, Toast.LENGTH_LONG).show()
52                 }
53             }
54         }
55
56         return view
57     }
58 }

```

```

1 package com.example.bqc
2
3 > import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 class QualityQuiz : AppCompatActivity() {
19     private lateinit var binding: ActivityQualityQuizBinding
20     private var goodCount = 0
21     private var averageCount = 0
22     private var badCount = 0
23     private val badMaterials = mutableListOf<String>()
24
25     override fun onCreate(savedInstanceState: Bundle?) {
26         super.onCreate(savedInstanceState)
27         setContentView(R.layout.activity_quality_quiz)
28
29         val resultTextView: TextView = findViewById(R.id.resultTextView)
30         val checkButton: Button = findViewById(R.id.checkQualityButton)
31         val db = Firebase.firestore
32         val documentID = intent.getStringExtra(name: "documentID")
33         checkButton.setOnClickListener {
34             // Reset all counters
35             goodCount = 0
36             averageCount = 0
37             badCount = 0
38             badMaterials.clear()
39
40             // Run evaluations
41             evaluate(
42                 findViewById(R.id.brickClassGroup),
43                 mapOf(
44                     R.id.brickOptionA to "Bad",
45                     R.id.brickOptionB to "Average",
46                     R.id.brickOptionC to "Good",
47                     R.id.brickOptionD to "Good"
48                 ),
49                 materialName: "Bricks"
50             )
51
52             evaluate(
53                 findViewById(R.id.steelGradeGroup),
54                 mapOf(
55                     R.id.steelOptionA to "Average",
56                     R.id.steelOptionB to "Good",
57                     R.id.steelOptionC to "Good"
58                 )
59             )
60         }
61     }
62 }

```

```

52         evaluate(
53             findViewById(R.id.steelGradeGroup),
54             mapOf(
55                 R.id.steelOptionA to "Average",
56                 R.id.steelOptionB to "Good",
57                 R.id.steelOptionC to "Good",
58                 R.id.steelOptionD to "Bad"
59             ),
60             materialName: "TMT Steel"
61         )
62     )
63
64     evaluate(
65         findViewById(R.id.concreteGradeGroup),
66         mapOf(
67             R.id.concreteOptionA to "Bad",
68             R.id.concreteOptionB to "Average",
69             R.id.concreteOptionC to "Good",
70             R.id.concreteOptionD to "Good"
71         ),
72         materialName: "Concrete"
73     )
74
75     evaluate(
76         findViewById(R.id.waterAbsorptionGroup),
77         mapOf(
78             R.id.gbsorpOptionA to "Bad",
79             R.id.gbsorpOptionB to "Average",
80             R.id.gbsorpOptionC to "Good",
81             R.id.gbsorpOptionD to "Good"
82         ),
83         materialName: "Brick Water Absorption"
84     )
85
86     evaluate(
87         findViewById(R.id.waterproofingGroup),
88         mapOf(
89             R.id.waterproofOptionA to "Bad",
90             R.id.waterproofOptionB to "Average",
91             R.id.waterproofOptionC to "Good",
92             R.id.waterproofOptionD to "Good"
93         )
94     )
95 }

```

```

93         materialName: "Waterproofing"
94     )
95
96     evaluate(
97         findViewById(R.id.wiringGroup),
98         mapOf(
99             R.id.wiringOptionA to "Bad",
100             R.id.wiringOptionB to "Average",
101             R.id.wiringOptionC to "Good",
102             R.id.wiringOptionD to "Good"
103         ),
104         materialName: "Electrical Wiring"
105     )
106
107     evaluate(
108         findViewById(R.id.fireRatingGroup),
109         mapOf(
110             R.id.fireOptionA to "Bad",
111             R.id.fireOptionB to "Average",
112             R.id.fireOptionC to "Average",
113             R.id.fireOptionD to "Good"
114         ),
115         materialName: "Fire Door Rating"
116     )
117
118     evaluate(
119         findViewById(R.id.plasterThicknessGroup),
120         mapOf(
121             R.id.plasterOptionA to "Bad",
122             R.id.plasterOptionB to "Average",
123             R.id.plasterOptionC to "Good",
124             R.id.plasterOptionD to "Good"
125         ),
126         materialName: "Plaster Thickness"
127     )
128
129     val result = buildResultReport()
130     resultTextView.text = result
131     Log.v("tag: debugApp", msg: "Changed result, now updating data")
132     if (documentID != null) {
133         db.collection("collectionPath: 'sites'").document("documentPath: documentID?::").update("field: 'qualityCheck', result).addOnCompleteListener {

```

```

138 class QualityQuiz : AppCompatActivity() {
139     override fun onCreate(savedInstanceState: Bundle?) {
140         checkButton.setOnClickListener {
141             if (documentID != null) {
142                 db.collection("collectionPath: 'sites'").document("documentPath: documentID?::").update("field: 'qualityCheck', result).addOnCompleteListener {
143                     Toast.makeText(context, this, text: "Updated quality check in the database", Toast.LENGTH_SHORT).show()
144                 }.addOnFailureListener {
145                     Toast.makeText(context, this, text: "Failed to update quality check in the database", Toast.LENGTH_SHORT).show()
146                 }
147             } else {
148                 Toast.makeText(context, this, text: "Invalid documentID", Toast.LENGTH_SHORT).show()
149             }
150         }
151     }
152
153     private fun evaluate(group: RadioGroup, answerKey: Map<Int, String>, materialName: String) {
154         val selectedId = group.checkedRadioButtonId
155         when (answerKey[selectedId]) {
156             "Good" -> goodCount++
157             "Average" -> averageCount++
158             "Bad" -> {
159                 badCount++
160                 badMaterials.add(materialName)
161             }
162         }
163     }
164
165     private fun buildResultReport(): String {
166         val total = goodCount + averageCount + badCount
167
168         val summary = when {
169             goodCount >= 6 -> "🟢 Most materials are GOOD. Construction quality looks solid!"
170             averageCount >= 4 -> "🟡 Several materials are AVERAGE. May need a few improvements."
171             badCount >= 2 -> "🔴 Warning: Multiple BAD materials detected. Please take action!"
172             else -> "🟢 Mostly acceptable, with a few areas to monitor."
173         }
174
175         val badList = if (badMaterials.isEmpty()) {
176             "\n\n🟢 Bad Materials Used:\n" + badMaterials.joinToString(separator = "\n ")
177         } else {
178             "\n\n🔴 No bad materials detected!"
179         }
180     }

```

```

167         val badList = if (badMaterials.isNotEmpty()) {
168             "\n\n🔴 Bad Materials Used:\n• " + badMaterials.joinToString(separator: "\n• ")
169         } else {
170             "\n\n🟢 No bad materials detected!"
171         }
172
173         return """
174             📋 Quality Check Report:
175             • Good: $goodCount
176             • Average: $averageCount
177             • Bad: $badCount
178             • Total Checks: $total
179
180             🏁 Verdict:
181             $summary
182
183             $badList
184             """.trimIndent()
185     }
186 }

```

## Report Issues

```

1 package com.example.bqc
2
3 import androidx.appcompat.app.AppCompatActivity
4 import androidx.appcompat.widget.Toolbar
5 import androidx.core.view.WindowCompat
6 import androidx.core.view.WindowInsetsCompat
7 import androidx.core.view.WindowInsetsControllerCompat
8 import androidx.recyclerview.widget.LinearLayoutManager
9 import androidx.recyclerview.widget.RecyclerView
10 import androidx.appcompat.widget.Toolbar
11 import androidx.appcompat.widget.Toolbar
12 import androidx.appcompat.widget.Toolbar
13 import androidx.appcompat.widget.Toolbar
14 import androidx.appcompat.widget.Toolbar
15
16 class ReportIssues : AppCompatActivity() {
17     private lateinit var binding: ActivityReportIssuesBinding
18     private lateinit var firebaseAuth: FirebaseAuth
19     override fun onCreate(savedInstanceState: Bundle?) {
20         super.onCreate(savedInstanceState)
21         enableEdgeToEdge()
22         binding = ActivityReportIssuesBinding.inflate(layoutInflater)
23         setContentView(binding.root)
24         val db = FirebaseFirestore.getInstance()
25         val firebaseAuth = FirebaseAuth.getInstance()
26         val documentId = intent.getStringExtra("documentID")
27         val categoryDropdown = binding.dropdownCategory
28         val descriptionInput = binding.inputDescription
29
30         val categories = arrayOf("App Crash", "Bug Report", "Payment Issue", "Other")
31         val adapter = ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, categories)
32         categoryDropdown.setAdapter(adapter)
33         binding.btnSubmit.setOnClickListener {
34             val category = categoryDropdown.text.toString().trim()
35             val description = descriptionInput.text.toString().trim()
36             if (category.isEmpty() || description.isEmpty()) {
37                 Toast.makeText(this, "Please fill in all fields", Toast.LENGTH_SHORT).show()
38             } else {
39                 db.collection("sites").document(documentId).update(
40                     mapOf("report" to "${firebaseAuth.currentUser?.email}'s report")
41                 )
42                 Toast.makeText(this, "Issue reported successfully!", Toast.LENGTH_LONG).show()
43                 finish()
44             }
45             binding.btnSubmit.setOnClickListener {
46                 Toast.makeText(this, "Failed to submit report", Toast.LENGTH_SHORT).show()
47                 finish()
48             }
49         }
50     }
51 }

```

## Site Actions

```

1 package com.example.bqc
2
3 import androidx.appcompat.app.AppCompatActivity
4 import androidx.appcompat.widget.Toolbar
5 import androidx.core.view.WindowCompat
6 import androidx.core.view.WindowInsetsCompat
7 import androidx.core.view.WindowInsetsControllerCompat
8 import androidx.recyclerview.widget.LinearLayoutManager
9 import androidx.recyclerview.widget.RecyclerView
10 import androidx.appcompat.widget.Toolbar
11 import androidx.appcompat.widget.Toolbar
12 import androidx.appcompat.widget.Toolbar
13 import androidx.appcompat.widget.Toolbar
14 import androidx.appcompat.widget.Toolbar
15
16 class SiteActions : AppCompatActivity() {
17     private lateinit var binding: ActivitySiteActionsBinding
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         enableEdgeToEdge()
21         binding = ActivitySiteActionsBinding.inflate(layoutInflater)
22         setContentView(binding.root)
23         val documentID = intent.getStringExtra("documentID")
24         binding.qualityCheck.setOnClickListener {
25             val i = Intent(packageContext, MainActivity::class.java)
26             i.putExtra("documentID", documentID)
27             startActivity(i)
28         }
29         binding.qualityCheck.setOnClickListener {
30             val i = Intent(packageContext, QualityQuiz::class.java)
31             i.putExtra("documentID", documentID)
32             startActivity(i)
33         }
34         binding.report.setOnClickListener {
35             val i = Intent(packageContext, ReportIssues::class.java)
36             i.putExtra("documentID", documentID)
37             startActivity(i)
38         }
39     }
40 }

```

## Site Details

```
1 package com.example.bgc
2
3 > import ...
4
15 class SiteDetails : AppCompatActivity() {
16     private lateinit var binding: ActivitySiteDetailsBinding
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         enableEdgeToEdge()
20         binding = ActivitySiteDetailsBinding.inflate(layoutInflater)
21         setContentView(binding.root)
22         val db = Firebase.firestore
23         val documentID = intent.getStringExtra("name: documentID")
24         if(documentID==null){
25             Toast.makeText(context: this, text: "null documentID", Toast.LENGTH_SHORT).show()
26             finish()
27         }
28         db.collection(collectionPath: "sites").document(documentPath: documentID?:"").get().addOnSuccessListener {
29             document->
30             if(document.exists()){
31                 binding.name.text = document.getString( field: "name")
32                 binding.description.text=document.getString( field: "description")
33                 binding.materialsUsed.text="Materials Used: "+document.getString( field: "materialsUsed")
34                 binding.siteTypeText.text="Site Type: "+ document.getString( field: "siteType")
35                 binding.currentWork.text="Current Work: "+document.getString( field: "currentWork")
36                 binding.totalBudget.text="Total Budget: "+document.getString( field: "totalBudget")
37                 binding.spendings.text="Spendings: "+document.getString( field: "spendings")
38                 binding.startDate.text="Start Date: "+document.getString( field: "startDate")
39                 binding.endDate.text="End Date: "+document.getString( field: "endDate")
40                 binding.siteStatusText.text="Site Status: "+document.getString( field: "siteStatus")
41                 binding.qualityDetails.text = document.getString( field: "qualityCheck")
42                 if(document.getString( field: "siteStatus")==="Active"){
43                     binding.siteStatusText.setTextColor(Color.parseColor( colorString: "#00FF00"))
44                 }else if(document.getString( field: "siteStatus")==="Completed"){
45                     binding.siteStatusText.setTextColor(Color.parseColor( colorString: "#0000FF"))
46                 }else if(document.getString( field: "siteStatus")==="To Start"){
47                     binding.siteStatusText.setTextColor(Color.parseColor( colorString: "#FFFF00"))
48                 }
49             }
50             }.addOnFailureListener {
51                 Toast.makeText(context: this, text: "Failed to fetch data", Toast.LENGTH_SHORT).show()
52                 finish()
53             }
54             binding.nextBtn.setOnClickListener {
55
56                 }
57                 }.addOnFailureListener {
58                     Toast.makeText(context: this, text: "Failed to fetch data", Toast.LENGTH_SHORT).show()
59                     finish()
60                 }
61                 binding.nextBtn.setOnClickListener {
62                     val i = Intent( packageContext: this, SiteActions::class.java)
63                     i.putExtra( name: "documentID",documentID)
64                     startActivity(i)
65                 }
66                 binding.updateBtn.setOnClickListener {
67                     val i = Intent( packageContext: this, AddSite::class.java)
68                     i.putExtra( name: "documentID",documentID)
69                     startActivity(i)
70                 }
71             }
72         }
73     }
74 }
```

## Site List

```
1 package com.example.bqc
2
3 > import
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 data class ConstructionSite(
23     val name: String,
24     val status: String,
25     val address: String,
26     val documentID: String
27 )
28
29
30
31 class SiteAdapter(private val siteList: List<ConstructionSite>) :
32     RecyclerView.Adapter<SiteAdapter.SiteViewHolder>() {
33
34     class SiteViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
35         val siteName: TextView = itemView.findViewById(R.id.textViewSiteName)
36         val siteStatus: TextView = itemView.findViewById(R.id.textViewStatus)
37         val siteAddress: TextView = itemView.findViewById(R.id.textViewAddress)
38     }
39
40     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): SiteViewHolder {
41         val view = LayoutInflater.from(parent.context)
42             .inflate(R.layout.item_site_card, parent, attachToRoot = false)
43         return SiteViewHolder(view)
44     }
45
46     override fun onBindViewHolder(holder: SiteViewHolder, position: Int) {
47         val site = siteList[position]
48         holder.siteName.text = site.name
49         holder.siteStatus.text = "Status: ${site.status}"
50         holder.siteAddress.text = site.address
51         if(site.status.toString()=="Active"){
52             holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_active_background)
53         }
54         if(site.status.toString()=="Completed"){
55             holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_completed_background)
56         }
57         if(site.status.toString()=="To start"){
58             holder.siteStatus.background=ContextCompat.getDrawable(holder.itemView.context,R.drawable.status_ongoing_background)
59         }
60
61         holder.itemView.setOnClickListener{
62             val context = holder.itemView.context
63             val intent = Intent(holder.itemView.context, SiteDetails::class.java)
64
65         }
66     }
67
68     override fun getItemCount(): Int = siteList.size
69 }
70
71
72 < > class SiteList : AppCompatActivity() {
73     private lateinit var recyclerView: RecyclerView
74     private lateinit var adapter: SiteAdapter
75     override fun onCreate(savedInstanceState: Bundle?) {
76         super.onCreate(savedInstanceState)
77         enableEdgeToEdge()
78         setContentView(R.layout.activity_site_list)
79         val siteList = mutableListOf<ConstructionSite>()
80         val db = FirebaseFirestore
81         setContentView(R.layout.activity_site_list)
82         recyclerView = findViewById<RecyclerView>(<R.id.recyclerView>)
83         recyclerView.layoutManager = LinearLayoutManager(<context> this)
84         adapter = SiteAdapter(siteList)
85         recyclerView.adapter = adapter
86         val loadingWheel = findViewById<ProgressBar>(<R.id.loadingWheel>)
87         db.collection(<collectionName> "sites").get().addOnSuccessListener {
88             querySnapshot->
89             for (document in querySnapshot){
90                 val documentId = document.id
91                 siteList.add(ConstructionSite( <name> document.getString( <field> "name"): "Unknown", <status> document.getString( <field> "siteStatus"): "Unknown", <address> document.getString( <field> "address"): "Unknown" ))
92             }
93             loadingWheel.visibility = View.INVISIBLE
94             adapter.notifyDataSetChanged()
95         }.addOnFailureListener {
96             Toast.makeText(<context> this, <text> "Failed to fetch data", Toast.LENGTH_SHORT).show()
97         }
98     }
99 }
```