2. Identify paths for 100 % statement and branch coverage.
3. Develop test data.

1. Login

Path1 : temp[] -> if(role=== ‘Academic staff”)(T) -> role= “Staff” -> if (role=== “Non Academic staff”)(T) -> role = “AR” -> query statements -> if (temp.length>0) (T) -> definitions -> if (temp.length=== 1 && role === “Student)(T) -> url= “/Student-Home” … return False -> return false

Path 2 : temp[] -> if(role=== ‘Academic staff”)(F) -> if (role=== “Non Academic staff”)(F) -> query statements -> if (temp.length>0) (F) -> return false

Path 3 : temp[] -> if(role=== ‘Academic staff”)(T) -> role= “Staff” -> if (role=== “Non Academic staff”)(T) -> role = “AR” -> query statements -> if (temp.length>0) (T) -> definitions -> if (temp.length=== 1 && role === “Student)(F) -> if (temp.length === 1 && role === “Staff”)(T) -> url= “Acc-Staff-Home .. return false -. Return false

Path 4 : temp[] -> if(role=== ‘Academic staff”)(T) -> role= “Staff” -> if (role=== “Non Academic staff”)(T) -> role = “AR” -> query statements -> if (temp.length>0) (T) -> definitions -> if (temp.length=== 1 && role === “Student)(F) -> if (temp.length === 1 && role === “Staff”)(F) -> if (temp.length === 1 && role === “AR”)(T) -> url = “/Non-Acc-Staff-Home”... return false -> return false

Path 4 : temp[] -> if(role=== ‘Academic staff”)(T) -> role= “Staff” -> if (role=== “Non Academic staff”)(T) -> role = “AR” -> query statements -> if (temp.length>0) (T) -> definitions -> if (temp.length=== 1 && role === “Student)(F) -> if (temp.length === 1 && role === “Staff”)(F) -> if (temp.length === 1 && role === “AR”)(F) -> return false

Test Cases:

Input	Expected Output
Role= Academic Staff	Url= “/Acc-Staff-Home”
Role= Non-Academic Staff	Url= “/Non-Acc-Staff-Home”
Role= Student	Url= “/Student-Home”
Role= Professor	return false

2. Upload Document

Path 1:
definitions-> i=0 -> i < approvers.length(T) -> if (i===0)(T) -> Processing -> comment.push(“”)-> i++ -> i < approvers.length(F) -> definitions

Path 2:
definitions-> i=0 -> i < approvers.length(T) -> if (i===0)(F) -> Waiting -> comment.push(“”)-> i++ -> i < approvers.length(F) -> definitions

Test Cases:

Input	Expected Output
No.of approvers= 1 , Documents*	status= waiting, comments= comments by approvers
No. of approvers =0	status= processing, comments=[]

3. Set document status

Path 1:
Definitions- > let i=0 -> if (i < docSnap.data().Approvers.length(T) -> if (docSnap.data().Approvers[i]=== staff && docSnap.data().Status[i] === “Processing)(T) -> definitions -> if (decision === “Approved”) (T) -> st[i]= “Approved” -> if (i+1 ===docSnap.data().Approvers.length)(T) -> tSt= “Approved” -> await UpdateDec -> i++ -> if (i < docSnap.data().Approvers.length(F) -> alert(Finished)

Path 2:
Definitions- > let i=0 -> if (i < docSnap.data().Approvers.length(T) -> if (docSnap.data().Approvers[i]=== staff && docSnap.data().Status[i] === “Processing)(F) -> i++ -> if (i < docSnap.data().Approvers.length(F) -> alert(Finished)

Path 3:
Definitions- > let i=0 -> if (i < docSnap.data().Approvers.length(T) -> if (docSnap.data().Approvers[i]=== staff && docSnap.data().Status[i] === “Processing)(T) -> definitions -> if (decision === “Approved”) (T) -> st[i]= “Approved” -> if (i+1

===docSnap.data().Approvers.length)(F) -> st[i+1] = “Processing” -> await UpdateDec -> i++ -> if (i < docSnap.data().Approvers.length(F) -> alert(Finished)

Path 4:
Definitions- > let i=0 -> if (i < docSnap.data().Approvers.length(T) -> if (docSnap.data().Approvers[i]=== staff && docSnap.data().Status[i] === “Processing)(T) -> definitions -> if (decision === “Approved”) (T)-> else if (decision === “Rejected)(T) -> st[i]= “Rejected” , tSt= “Rejected” -> await UpdateDec -> i++ -> if (i < docSnap.data().Approvers.length(F) -> alert(Finished)

Path 5:
Definitions- > let i=0 -> if (i < docSnap.data().Approvers.length(T) -> if (docSnap.data().Approvers[i]=== staff && docSnap.data().Status[i] === “Processing)(T) -> definitions -> if (decision === “Approved”) (T)-> else if (decision === “Rejected)(F)-> await UpdateDec -> i++ -> if (i < docSnap.data().Approvers.length(F) -> alert(Finished)

Test Cases:

Input	Decision by approver	Expected Output
In docSnapdata , approvers =1 , status = processing , approver[0]= staff	Approved	St[i] = “Approved” , tSt= Approved alert
In docSnapdata , approvers =1 , status = waiting , approver[0]= staff	-	status= nill, alert
In docSnapdata , approvers =2 , status = processing , approver[0]= staff	approved	st[0]= approved, st[1]= processing , alert
In docSnapdata , approvers =1 , status = processing , approver[0]= staff	Rejected	St[0] = Rejected, tSt= Rejected , alert
In docSnapdata , approvers =1 , status = processing , approver[0]= staff	checking	st[0]= nill, alert

4. Read Documents Staff

Path 1:
definitions- > snapshot -> i=0 -> i< doc.data().Approvers.length(T) -> if (doc.data().Approvers[i]=== staff && doc.data().Status[i]= status)(T)-> temp.push(doc.data())-> i++ -> i< doc.data().Approvers.length(F)-> return temp

Path 2:
definitions- > snapshot -> i=0 -> i< doc.data().Approvers.length(T) -> if (doc.data().Approvers[i]=== staff && doc.data().Status[i]= status)(F)-> i++ -> i< doc.data().Approvers.length(F)-> return temp

Test cases:

Input	Expected Output
approvers= 1 , approvers[0]= staff , status[0]= status	Document data pushed*
approvers= 1 , approvers[0]= student , status[0]= status	Document not pushed

5. Get Path Flow Details

Path 1:
definitions -> docSnap-> if (docSnap.exists())(T) -> definitions -> return details

- No path for false statement was given in the code, which represents a bug in the code

Test cases:

Input	Expected Output
docSnap	Details of document data and approver of the document

6. Upload Path details
Path 1:
Pathref -> await setDoc -> return true

Test Case:

Input	Expected Output
Path to document	Push path document, true

7. Read Rejected Document

Path 1:
Query to db-> snapshot -> temp.push(Doc.data()) -> return temp

Test Case:

Input	Expected Output
Requester mail= email@email.com , file_status= rejected	Document data is pushed for the viewer to read*