Dan Belair (bela0019)
Brandon Chan (chanx411)
Nic Hamlin (hamli052)
Keenan Perera (perer026)

Testing Synopsis (Team 28)

Summary

In general, our tests confirmed that our database functions well. We ran into several minor flaws with the design, the majority of which are correctable through programming. However, one major error we noticed was that patient has a foreign key to guardian when in reality, it should be reversed — this a problem with the design. Our test results, along with specifics about the errors we found, are summarized in more detail below.

Test 1:

Our first test confirmed that, when they are created, all patients in our system are required to have a guardian associated with them. This fits the requirement in the project description that patients under the age of 18 must have a guardian. Because *all* patients required a guardian, this led to errors when creating patients that are over the age of 18 (more detail is in the Test 2 summary).

Test 2:

Our current design does not capture this requirement. The database design has Patient's PID attribute having a foreign-key relationship with Guardian's PatID attribute. This is a flaw in our design — it should have been the other way around, with Guardian.PatID acting as a foreign key to Patient.PID. In its current design, all patients must have a guardian, no matter their age. There is a possible workaround of making patients over 18 their own guardian, but this is not an ideal solution.

Test 3:

This system does document when a Patient does not have Insurance. We tested this by creating a new Payment Method *without* an Insurance table referencing it as a foreign key, then queried that Payment Method. It displayed as empty. At the moment, there is no requirement for a Self Pay to be introduced if a Patient does not have Insurance. This could be corrected with an assertion that a Patient must have a Payment Method (an Insurance without an "end date" or a Self Pay) on file.

Test 4:

Our system is designed to collect old insurance information through the use of Insurance's "start date" and "end date" attributes. We tested this by adding a new insurance to a patient, then selecting all of the insurance(s) stored for that patient. One possible flaw is that, when an Insurance is added, the old Insurance is not required to have its "end date" updated. This is

correctable through a trigger, set to require that an “end date” for old Insurance be added whenever a new Insurance is added.

Test 5:

A Patient, as required, can be visited by one or more service providers. We added multiple service providers and multiple diagnoses to test this, and the system did not throw any errors. Similar to in Test 6, we noted that even if a patient is seen by multiple service providers, it stays within the scope of a single Visit (and thus a single Bill).

Test 6:

A Patient, as required, can have more than one diagnosis per visit To test this, we added multiple diagnoses (both from the same and from different service providers) to the same patient, and queried to make sure they didn’t overwrite each other. All diagnoses saved and were queried successfully, showing that a patient can have more than one diagnosis (as intended) from any combination of service providers. As an added design benefit, because this is in the scope of a single Visit, no matter how many diagnoses the patient receives, there is only one Bill.

Test 7:

We tested whether non-service providers could input diagnoses. After testing multiple Nurses, only those who were service providers could input diagnoses. In order for a diagnosis to be allowed, the service provider ID in diagnosis must also be in Service Provider (via FK -> PK relation).

Test 8:

Proper followup procedures are documented by a clerk in our system. The design was set up so a patient can have a visit and have recurring visits/check ups. The database properly adds the follow ups to the database, including when payment details are modified. This is consistent with the requirement that followup visits are properly documented. One possible issue with this setup is that the follow ups are currently assumed to be a part of the same Visit, when it’s possible that the patient will get follow up procedures done in a different visit.

Test 9:

The system records which clerk documents the Payment Method through the ClerkID attribute. An intake clerk records all payment data, and that is reflected in the payment method. We tested this by adding multiple clerks, and checking that the Payment Method correctly identified which clerk updated it.

Test 10:

To test whether the system allows for more than one Initial Assessment at a time, we added another initial assessment. It added without tossing an error, indicating a flaw in our design. To correct this, we would need to update the design for Initial Assessment's relationship to VisitID — it's Assessment ID attribute should have a foreign key to Visit.VisitID to ensure a 1-1 relationship.

Test 11:

To test whether a Patient can have multiple Visits, we attempted to add another Visit with a Patient ID that is already in the system. There are no restrictions on Visit's Patient ID, so another Visit was created without issues for the same Patient.

Test 12:

All initial assessments have a Nurse ID attribute, however this attribute is not specified as NOT NULL or <0, so the potential exists for a Nurse ID to not be filled out for a particular initial assessment. That information is retrievable, however, through the Nurse ID as a foreign key to Nurse.NurseID, which is a foreign key to Employee.EmpIID (the table in which the nurse's information is stored).

Test 13:

From our testing, we show that the system is capable of a single Nurse performing (and recording) Initial Assessments for multiple patients. This is expected in the Clinic environment and as it should be according to the project specifications.

Test 14:

Our system only allows for one Nurse to be added to an Initial Assessment. The error "ERROR Code:1062 Duplicate Entry" appears if an attempt is made to add a second Nurse into the Initial Assessment table. Since only one Nurse is supposed to be completing any Initial Assessment, our system accomplishes this requirement and passes the test.

Test 15:

Vitals and Symptoms are both attributes of the Initial Assessment and can be reviewed through a select query on the table. This does not query the Visit necessarily, but will be intrinsically linked through our use of foreign and primary keys.

Test 16:

We accomplish this criteria with a pseudo-boolean value on our Employee. The attribute SalaryOrHourly functions to differentiate employees from one wage type to another. The

attribute is simply a varchar. So, the declaration of new employees must follow strict guidelines to ensure that the attribute is filed correctly. We can ensure this to be working with the addition of an Assertion of the SalaryOrHourly attribute which only allows for two types.

Test 17:

Our system does not account for the functionality of requesting tests to be done based on the gathered vital and symptom information. Therefore, doctors in our system are able to provide a treatment based on the diagnosis, but not able to request other forms of information-gain tests. This is a lapse in our design and in another iteration of the system design, we would need to ensure more robust policies.

Test 18:

Since the default patient will begin their journey in the hospital system without treatments given, they do begin visits without any Treatment on file. There is no requirement or trigger or assertion in our system to require this default status to change.

Test 19:

Test 20: