Amogh Bhagwat: bhagw018

Grant Coletstock: coles042

David Nguyen: nguy3482

Tony Phan: phan0143

Team #25

## New Haven Urgent Care Design Review

## Introduction/Overview

Overall, our design was mostly sound, requiring only minor modifications before being fully functional. Most of the requirements were met using only the table design and most of the remaining requirements could be met by programming in a few basic constraints. The largest flaw in our design is that a patient can be seen by multiple service providers, each giving a diagnosis. To fix this, we recommend having the visit record reference the service provider. Otherwise, we were able to capture the New Haven Urgent Care’s system well.

## Patient Design

First we will be focusing on if our design captured the requirements for patients, namely guardians and insurance. Without additional constraints, the current database does not capture the parent/guardian requirements for minors. Testing showed that patients who are under the age of 18 can make a visit without a guardian. This is because guardians are tracked per visit using a foreign key, and there is nothing enforcing the system to create a guardian when a minor makes a visit. It is not ideal that guardians need a pre-existing visit to be created in the first place.

However, this problem can be solved without too much hassle. By adding an assertion on the patient’s age and a trigger upon a visit creation, we can force the creation of a guardian for a minor’s visit. Since a little programming can handle this problem, our design is acceptable, as no major restructuring is necessary.

The requirement that legal adults don’t have guardians is mostly captured by our design but additional programming is needed to enforce this. Since guardians are tracked on a per visit basis, legal adults who were patients when they were under 18 won’t have their past (former) guardians tracked in their future visits. However, our testing showed there is nothing stopping us from associating a guardian with a currently legal adult. Our database would need an assertion to check the patient’s age at the time of visit before a guardian is created. This would prevent this issue of legal adults having guardians. Overall, this design worked fairly well for capturing age requirements, as full functionality is possible without major modifications on the patient table design.

Moving on from guardians, our testing showed that we fully captured insurance requirements regarding patients. The insurance policy of each patient is recorded as a foreign key in visit, which worked really well for tracking insurance policies between visits. The case when a patient who had insurance for one visit but no insurance for another visit was taken care since each visit had its own insurance policy value. While this means unchanging policies need to be recorded for each visit, this design handles changing insurance information well.

It’s important to note that in our design, a NULL insurance policy was treated as having no insurance.This is an area for improvement, as a NULL has ambiguity in its meaning and can be created erroneously. Instead of treating a NULL insurance policy value as no insurance, we should’ve used a special policy number to remove ambiguity.

## Diagnosis Design

The next section of requirements centers around the Diagnosis entity. There are both flaws and strengths around our design. The first flaw is that patients can be seen by multiple service providers during a visit. Patients can even have multiple service providers who give different diagnoses, which is not good. This can be fixed by including a key for a service provider in the visit table and foreign keying it to the primary key of the service provider table. This will ensure that only one diagnosis can be made per visit. A second flaw is that every visit can have no diagnoses. In other words, a visit can be inserted into the database without the existence of a diagnosis. In the real world, this may be interpreted as a visit with a NULL diagnosis means that there is nothing wrong with the patient. However, nothing is wrong with the patient could also be a diagnosis itself. In order to fulfill this requirement, a trigger could be added to visit where after inserting a visit, a diagnosis must also be inserted. Besides the two minor flaws, other aspects and requirements of the diagnosis seem to work well and as intended. Patients can be both seen by different service providers on different visits as well as patients can have multiple diagnoses for each individual visit. This is handled because each of these entities have their own tables with foreign keys that relate to each other. Overall, the design works and both of the flaws mentioned are relatively easy to fix and do not really break/require major rework of our original design. One is an addition to the ER/EER diagram/table mapping and the other is a code addition.

Besides the minor flaws in diagnosis design, there is also a point of ambiguity. In the requirement that states “can only service providers make diagnoses(es),” in short, this requirement is fulfilled because a diagnosis requires as a foreign key the id of the service provider who made it. However, because the service provider is also an employee, this is where the ambiguity comes in. Because employee is a prerequisite for both in clerk and service provider (as well as nurse, but nurse is not relevant in this test,) there is a situation where an employee could be created, then an in clerk off of that employee, and then a service provider off of that same employee. So this employee is both an in clerk and a service provider. The tests show that a service provider that is also an in clerk can make a diagnosis. The requirement is fulfilled that an employee who is a service provider makes the diagnosis, but does the fact that the service provider is also an in clerk invalidate the requirement? In practice, this may manifest as a clinic that does not have many employees available so the service provider must also act as the in clerk.

## Treatment Design

The next primary aspect of the design centered on ordering treatments (either tests or procedures) for a patient. First, we will examine the strengths of the chosen design and then explore how the design could be improved. One strength of the chosen design is that treatments are stored independently of visits. This means that the name and cost of a treat only needs to be input once and can be reused across multiple visits. Another strength of the design is that patients can be prescribed multiple treatments for each visit they make or no treatments depending on their needs. If the patient is a minor, each treatment could be approved separately.

Our design also has room for improvement. There are two issues which we will identify here. The first is that we cannot identify which service provider ordered a particular treatment. In our current design, the visit may have multiple diagnoses, each issued by a different doctor. As a result, there is not a single doctor for the visit and we cannot know who ordered each treatment for the patient. We would remedy this issue by relating each visit with a service provider so that they only had one doctor issuing diagnoses or treatments. The second issue we identified in our design was that a prescription for a minor can be inserted without parental approval. There is no check verifying that every treatment for a minor is approved. We recommend fixing this issue by adding a constraint which asserts that every prescription inserted for a minor has an approval form. In conclusion, the design relating treatments and visits was mostly sound, requiring only minor modifications to be perfectly functional.

## Visit Design

We now turn our attention to the design of visits in our system. Again we will focus first on the strengths of the design and then on the weaknesses. A patient is able to make multiple visits. For each visit, at most one initial assessment is taken. Further, the in clerk which checks the patient in is recorded on the visit record. A weakness in our design is that we cannot ensure that every visit has an initial assessment. We do not propose a fix for this because the visit record must be created before the initial assessment record. A second weakness in our visit design is that we record incomplete insurance information. We recognize that the patient pays some money upfront and the remaining balance is paid later, but we do not explicitly record the copay with the visit.

## Initial Assessment Design

Lastly our design of initial assessment worked for the tests that were presented to us. The way initial assessment is designed we store the nurses id as a foreign key, which means we can reference the id and get to the nurses information for any assessment we need. This choice of including nurse id to initial assessment also helped with the test case of nurses being able to fill out multiple patients initial assessments. If we had nurses id part of initial assessments primary key we wouldn’t be able to do this. This leads to our primary key set up for initial assessment. We decided to make an assessment id the primary key, which helped us with the test case for an initial assessment being completed by only one nurse. Our design passed this test because of the constraint on primary key and also having only one nurse id attribute. Two nurses can’t be entered into the system with the same assessment id because primary keys require uniqueness. The last test for initial assessment was to see if our design would let us get the patient's vitals given only a visit. Our design passed this test as well because of the decision to include a visit id as a foreign key in initial assessment. This means we can join the visit and initial assessment tables and retrieve the vitals of any patient we need without too much trouble.

## Conclusion

In conclusion, our design was mostly sound, requiring only minor modifications. Most of the remaining requirements could be addressed using triggers and assertions, such as adding age checks to determine if a guardian is needed. There are only minor changes required in the table design to meet all the requirements. For example, our design was flawed in how it captured the service provider for a certain visit in the diagnosis table. If we had instead captured the seen service provider directly in the visit table, we could’ve captured the requirement that each patient sees only 1 service provider per visit. Currently there is some ambiguity in our design for capturing certain things like which service provider prescribed what treatment and the fact that we used NULL in some areas to represent a special value. Overall, our design was solid. There are some flaws and adjustments must be made to the design to fully meet New Haven Urgent Care’s requirements. Thankfully, most of these changes are minor and can be implemented quickly.