**COMP-421 Project Phase 3**

**Group 12**

**March 2015**

Andrew Fogarty 260535895

Lillio Mok 260456786

Zheng (Mason) Sun 260464047

**1)**

**-- This table keeps records of patients who were treated successfully with the illness, the day**

**-- discharged, and the staff member treating the patient.**

*CREATE TABLE OutPatientTreatment (*

*patient\_id INTEGER,*

*ill\_id char(8),*

*treated\_by\_staff\_id INTEGER,*

*ill\_until DATE NOT NULL,*

*FOREIGN KEY (patient\_id)*

*REFERENCES Patient(patient\_id) ON DELETE CASCADE,*

*FOREIGN KEY (ill\_id)  
 REFERENCES Illness(ill\_id),*

*FOREIGN KEY (treated\_by\_staff\_id)*

*REFERENCES Staff(staff\_id) ON DELETE CASCADE,*

*PRIMARY KEY (patient\_id, ill\_id, ill\_until)*

*);*

**-- Procedure to log outpatient services for cured patient**

**-- Find most recent staff member to treat patient; note that a patient who hasn’t been treated and**

**-- has a null ill\_until date may not be deleted or considered to be an outpatient.**

*CREATE OR REPLACE FUNCTION log\_out\_patient() RETURNS TRIGGER AS $log\_out\_patient$*

*BEGIN*

*IF old.ill\_until IS NULL THEN*

*RAISE EXCEPTION 'ill\_until cannot be null';*

*END IF;*

*INSERT*

*INTO OutPatientTreatment(patient\_id, ill\_id, treated\_by\_staff\_id, ill\_until)*

*SELECT old.patient\_id, old.ill\_id, staff\_id, old.ill\_until*

*FROM Treats*

*WHERE Treats.patient\_id = old.patient\_id*

*ORDER BY Treats.since DESC*

*LIMIT 1;*

*RETURN old;*

*END;*

*$log\_out\_patient$ LANGUAGE 'plpgsql';*

**-- Response:**

CS421=> CREATE OR REPLACE FUNCTION log\_out\_patient() RETURNS TRIGGER AS $log\_out\_patient$

CS421$> BEGIN

CS421$> IF old.ill\_until IS NULL THEN

CS421$> RAISE EXCEPTION 'ill\_until cannot be null';

CS421$> END IF;

CS421$>

CS421$> INSERT

CS421$> INTO OutPatientTreatment(patient\_id, ill\_id, treated\_by\_staff\_id, ill\_until)

CS421$> SELECT old.patient\_id, old.ill\_id, staff\_id, old.ill\_until

CS421$> FROM Treats

CS421$> WHERE Treats.patient\_id = old.patient\_id

CS421$> ORDER BY Treats.since DESC

CS421$> LIMIT 1;

CS421$>

CS421$> RETURN old;

CS421$> END;

CS421$> $log\_out\_patient$ LANGUAGE 'plpgsql';

CREATE TRICREATE FUNCTION

**-- Trigger the procedure if we delete a patient**

*CREATE TRIGGER logOutPatient AFTER DELETE ON SufferingFrom*

*FOR EACH ROW EXECUTE PROCEDURE log\_out\_patient();*

**-- Response**

CS421=> CREATE TRIGGER logOutPatient AFTER DELETE ON SufferingFrom

CS421-> FOR EACH ROW EXECUTE PROCEDURE log\_out\_patient();

CREATE TRIGGER

**-- Examples; insert these if they are missing in the database**

*INSERT INTO SufferingFrom VALUES (691123, 'i0023325', '2014-05-15', '2014-06-01', 2000.00, 200.00, null);*

*INSERT INTO SufferingFrom VALUES (690000, 'i0023358', '2013-02-03', null, 60.00, 120.00, null);*

**-- Will not trigger because of null value**

CS421=> DELETE FROM SufferingFrom WHERE patient\_id = 690000 AND ill\_id = 'i0023358';

ERROR: ill\_until cannot be null

**-- Will trigger**

CS421=> DELETE FROM SufferingFrom WHERE patient\_id = 691123 AND ill\_id = 'i0023325';

DELETE 1

CS421=> SELECT \* FROM OutPatientTreatment;

patient\_id | ill\_id | treated\_by\_staff\_id | ill\_until

------------+----------+---------------------+------------

690001 | i0023358 | 34002 | 2013-02-04

691123 | i0023325 | 34001 | 2014-06-01

**2)**

**-- This procedure logs high priority cases and recommends doctors to be assigned to each patient**

**-- It first creates table HighPriorityPatient if it’s not already been created**

**-- It takes an argument type::VARCHAR(9) that may either be some version of ‘urgency’ or ‘infectious’**

**-- If urgency is chosen, patients of a high urgency will be considered to be high priority**

**-- If infectious is chosen, patients with infectious diseases will be considered to be high priority**

**-- It then polls a cursor created depending on the switches, and inserts the correct patients**

**-- Both require that the patient has not been treated yet (i.e. ill\_until date is NULL)**

**-- It returns an integer of the number of inserted high priority cases**

*CREATE OR REPLACE FUNCTION log\_high\_priority\_patients(type VARCHAR(9)) RETURNS INTEGER AS $log\_high\_priority\_patients$*

*DECLARE*

***-- These are all the variables we need, with patient\_id, ill\_id, ill\_since, urgency renamed***

*cur refcursor;*

*pat\_id INTEGER;*

*il\_id CHAR(8);*

*il\_since DATE;*

*urge VARCHAR(32);*

*count INTEGER := 0;*

*BEGIN*

***-- Try to create the new relation, if it exists then move on.***

*BEGIN*

*CREATE TABLE HighPriorityPatient(*

*patient\_id INTEGER,*

*ill\_id char(8),*

*ill\_since DATE,*

*recommended\_staff\_id INTEGER,*

*urgency VARCHAR(32),*

*contagious BOOLEAN,*

*FOREIGN KEY (patient\_id)*

*REFERENCES Patient(patient\_id) ON DELETE CASCADE,*

*FOREIGN KEY (ill\_id)*

*REFERENCES Illness(ill\_id),*

*FOREIGN KEY (recommended\_staff\_id)*

*REFERENCES Staff(staff\_id),*

*PRIMARY KEY (patient\_id, ill\_id, ill\_since)*

*);*

*EXCEPTION WHEN duplicate\_table THEN*

*END;*

***-- Case switches for the choice of high prioritisation based on urgency or infectiousness***

*CASE*

*WHEN type like '%urgency%' OR type like '%Urgency%' THEN*

*OPEN cur FOR SELECT SufferingFrom.patient\_id, SufferingFrom.ill\_id, SufferingFrom.ill\_since, SufferingFrom.urgency FROM SufferingFrom WHERE (urgency LIKE '%Urgent%' OR urgency like '%urgent%' AND ill\_until IS NULL);*

*WHEN type like '%infectious%' OR type like '%Infectious%' OR type like '%contagious%' OR type like '%Contagious%' THEN*

*OPEN cur FOR SELECT patient\_id, Illness.ill\_id, ill\_since, urgency FROM Illness INNER JOIN SufferingFrom ON Illness.ill\_id = SufferingFrom.ill\_id WHERE contagious = TRUE AND ill\_until IS NULL;*

*ELSE*

*RAISE EXCEPTION 'Inapplicable high priority type.';*

*END CASE;*

***-- Look through the cursor***

*LOOP*

*FETCH cur INTO pat\_id, il\_id, il\_since, urge;*

*IF NOT FOUND THEN*

*EXIT;*

*END IF;*

***-- Insert the new tuple, but ignore it if it already exists w.r.t. its primary key***

*BEGIN*

*INSERT INTO HighPriorityPatient*

*SELECT pat\_id, il\_id, il\_since, NULL, urge, Illness.contagious*

*FROM Illness*

*WHERE Illness.ill\_id = il\_id;*

***-- attempt to find a suitable doctor***

*UPDATE HighPriorityPatient*

*SET recommended\_staff\_id = SpecializesIn.staff\_id*

*FROM SpecializesIn*

*WHERE SpecializesIn.ill\_id = il\_id*

*AND HighPriorityPatient.patient\_id = pat\_id*

*AND HighPriorityPatient.ill\_id = il\_id*

*AND HighPriorityPatient.ill\_since = il\_since;*

*count = count + 1;*

*EXCEPTION WHEN unique\_violation THEN*

*END;*

*END LOOP;*

*RETURN count;*

*END;*

*$log\_high\_priority\_patients$ LANGUAGE 'plpgsql';*

**-- Applicable patients:**

CS421=> SELECT \* FROM SUFFERINGFROM where urgency like '%urgent%' or urgency like '%Urgent%' AND ill\_until is NULL;

patient\_id | ill\_id | ill\_since | ill\_until | insurance\_coverage | treatment\_cost | urgency

------------+----------+------------+-----------+--------------------+----------------+------------------------

691126 | i0023313 | 2014-11-08 | | 2500.00 | 200.00 | Urgent

691128 | i0023313 | 2014-11-23 | | 15000.00 | 200.00 | Urgent

691129 | i0023366 | 2012-10-10 | | 5000.00 | 400.00 | very urgent

691129 | i0023378 | 2012-01-01 | | 10000.00 | 500.00 | very urgent I will die

(4 rows)

CS421=> SELECT \* FROM SUFFERINGFROM INNER JOIN ILLNESS ON SUFFERINGFROM.ILL\_ID = ILLNESS.ILL\_ID WHERE contagious = true AND ill\_until is NULL;

patient\_id | ill\_id | ill\_since | ill\_until | insurance\_coverage | treatment\_cost | urgency | ill\_id | ill\_name | contagious | average\_treatment\_cost

------------+----------+------------+-----------+--------------------+----------------+---------+----------+-------------------------------------------------+------------+------------------------

690001 | i0023358 | 2013-02-03 | | 60.00 | 60.00 | | i0023358 | unclassified disorders of the trigeminal nerves | t | 635.90

690000 | i0023358 | 2013-02-03 | | 60.00 | 120.00 | | i0023358 | unclassified disorders of the trigeminal nerves | t | 635.90

691128 | i0023313 | 2014-11-23 | | 15000.00 | 200.00 | Urgent | i0023313 | Gastroenteritis | t | 80.50

691127 | i0023313 | 2014-12-10 | | 2000.00 | 200.00 | Minor | i0023313 | Gastroenteritis | t | 80.50

691126 | i0023313 | 2014-11-08 | | 2500.00 | 200.00 | Urgent | i0023313 | Gastroenteritis | t | 80.50

691125 | i0023313 | 2014-12-07 | | 1000.00 | 200.00 | | i0023313 | Gastroenteritis | t | 80.50

691123 | i0023313 | 2015-03-07 | | 2000.00 | 200.00 | | i0023313 | Gastroenteritis | t | 80.50

(7 rows)

**-- To show that HighPriorityPatient does not exist**

CS421=> SELECT \* FROM HighPriorityPatient;

ERROR: relation "highprioritypatient" does not exist

LINE 1: SELECT \* FROM HighPriorityPatient;

**-- Responses after executing procedure**

CS421=> SELECT log\_high\_priority\_patients('infectious');

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "highprioritypatient\_pkey" for table "highprioritypatient"

CONTEXT: SQL statement "CREATE TABLE HighPriorityPatient(

patient\_id INTEGER,

ill\_id char(8),

ill\_since DATE,

recommended\_staff\_id INTEGER,

urgency VARCHAR(32),

contagious BOOLEAN,

FOREIGN KEY (patient\_id)

REFERENCES Patient(patient\_id) ON DELETE CASCADE,

FOREIGN KEY (ill\_id)

REFERENCES Illness(ill\_id),

FOREIGN KEY (recommended\_staff\_id)

REFERENCES Staff(staff\_id),

PRIMARY KEY (patient\_id, ill\_id, ill\_since)

)"

PL/pgSQL function "log\_high\_priority\_patients" line 12 at SQL statement

log\_high\_priority\_patients

----------------------------

7

(1 row)

CS421=> SELECT log\_high\_priority\_patients('urgency');

log\_high\_priority\_patients

----------------------------

2

(1 row)

CS421=> SELECT \* FROM HighPriorityPatient;

patient\_id | ill\_id | ill\_since | recommended\_staff\_id | urgency | contagious

------------+----------+------------+----------------------+------------------------+------------

690000 | i0023358 | 2013-02-03 | | | t

691123 | i0023313 | 2015-03-07 | 34001 | | t

691125 | i0023313 | 2014-12-07 | 34001 | | t

691126 | i0023313 | 2014-11-08 | 34001 | Urgent | t

691127 | i0023313 | 2014-12-10 | 34001 | Minor | t

691128 | i0023313 | 2014-11-23 | 34001 | Urgent | t

690001 | i0023358 | 2013-02-03 | | | t

691129 | i0023366 | 2012-10-10 | 34001 | very urgent | f

691129 | i0023378 | 2012-01-01 | 34091 | very urgent I will die | f

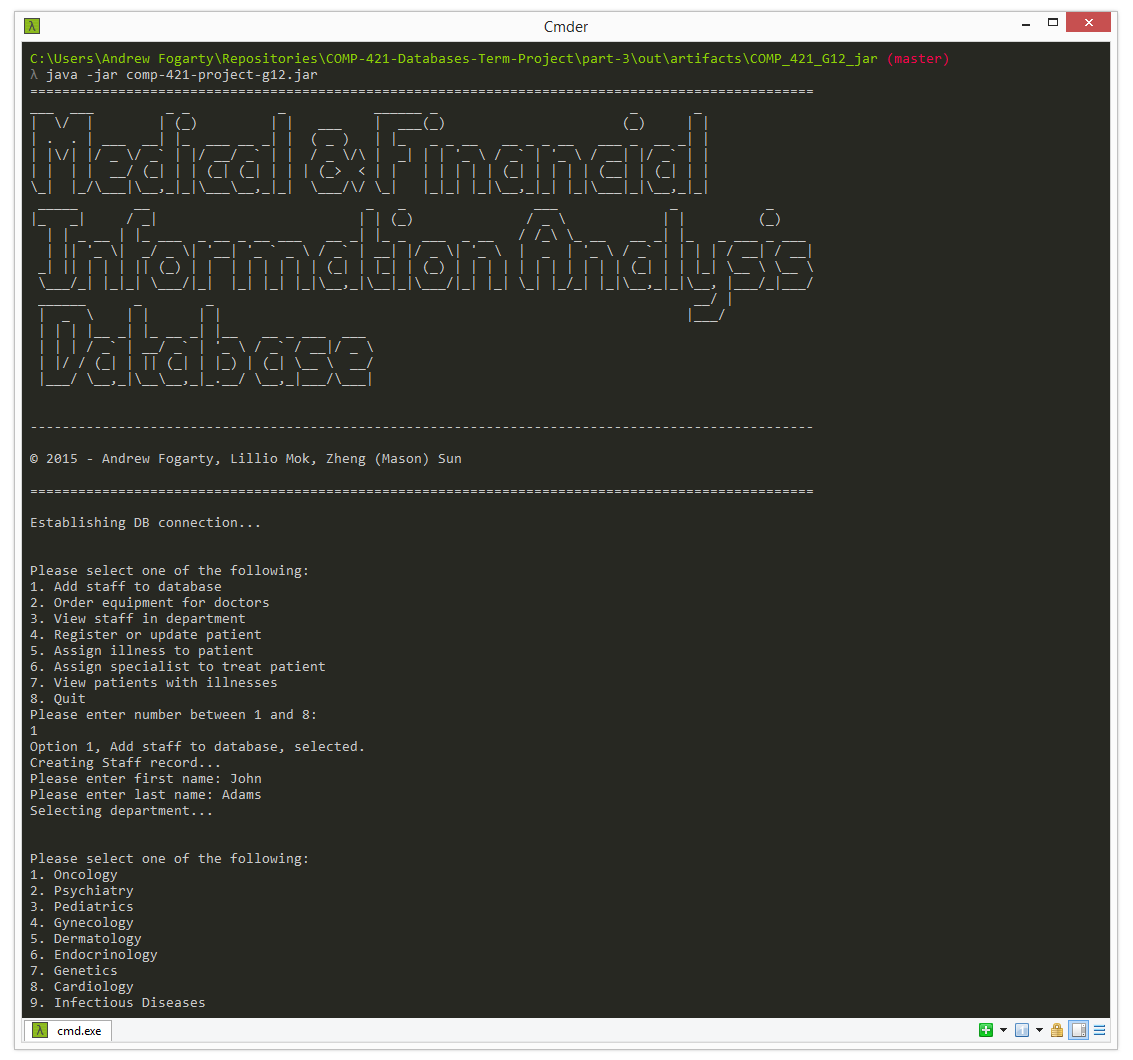
(9 rows)

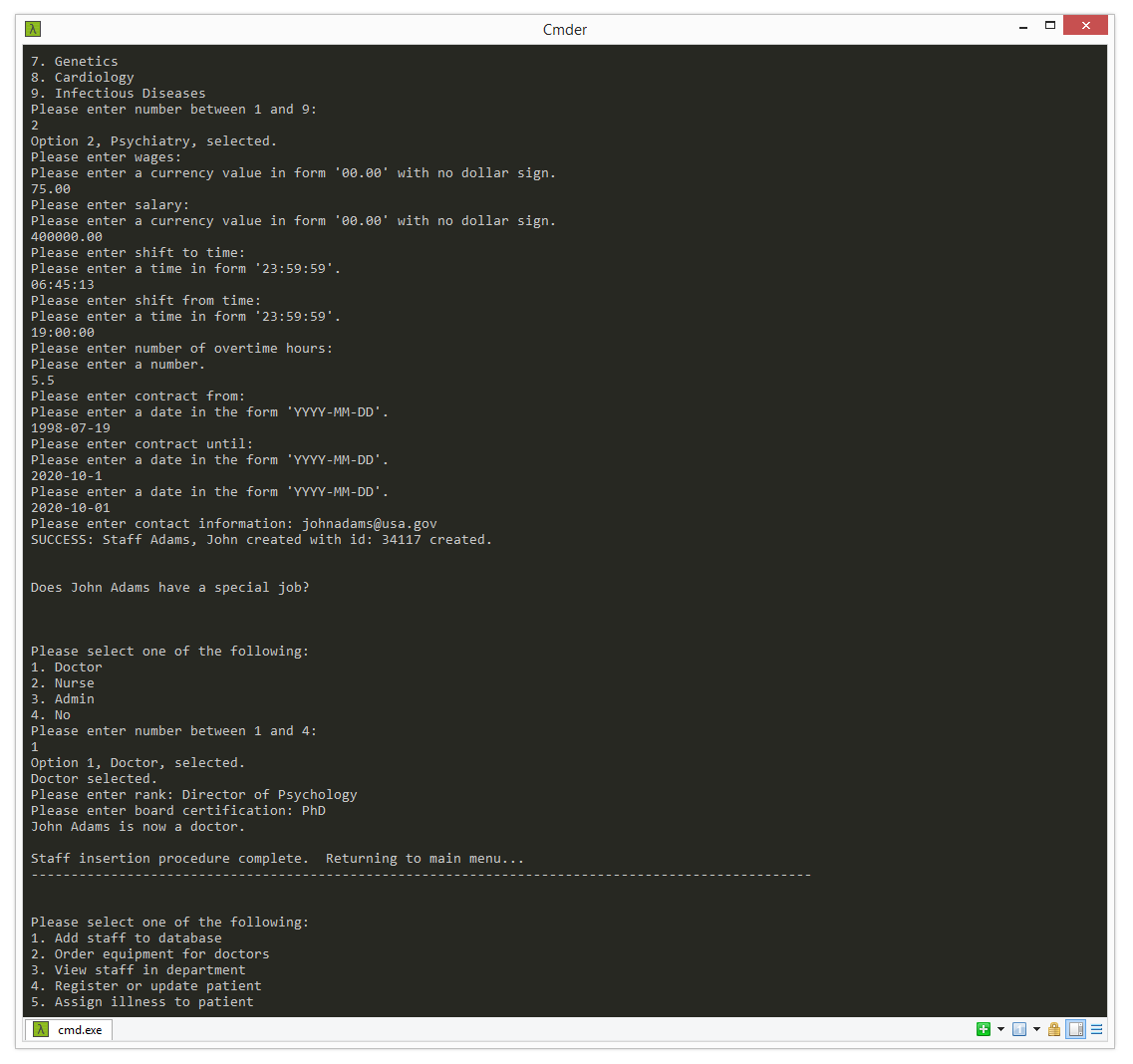
**-- Note that repeated patients just imply that they have multiple untreated conditions requiring attention**

**3)**

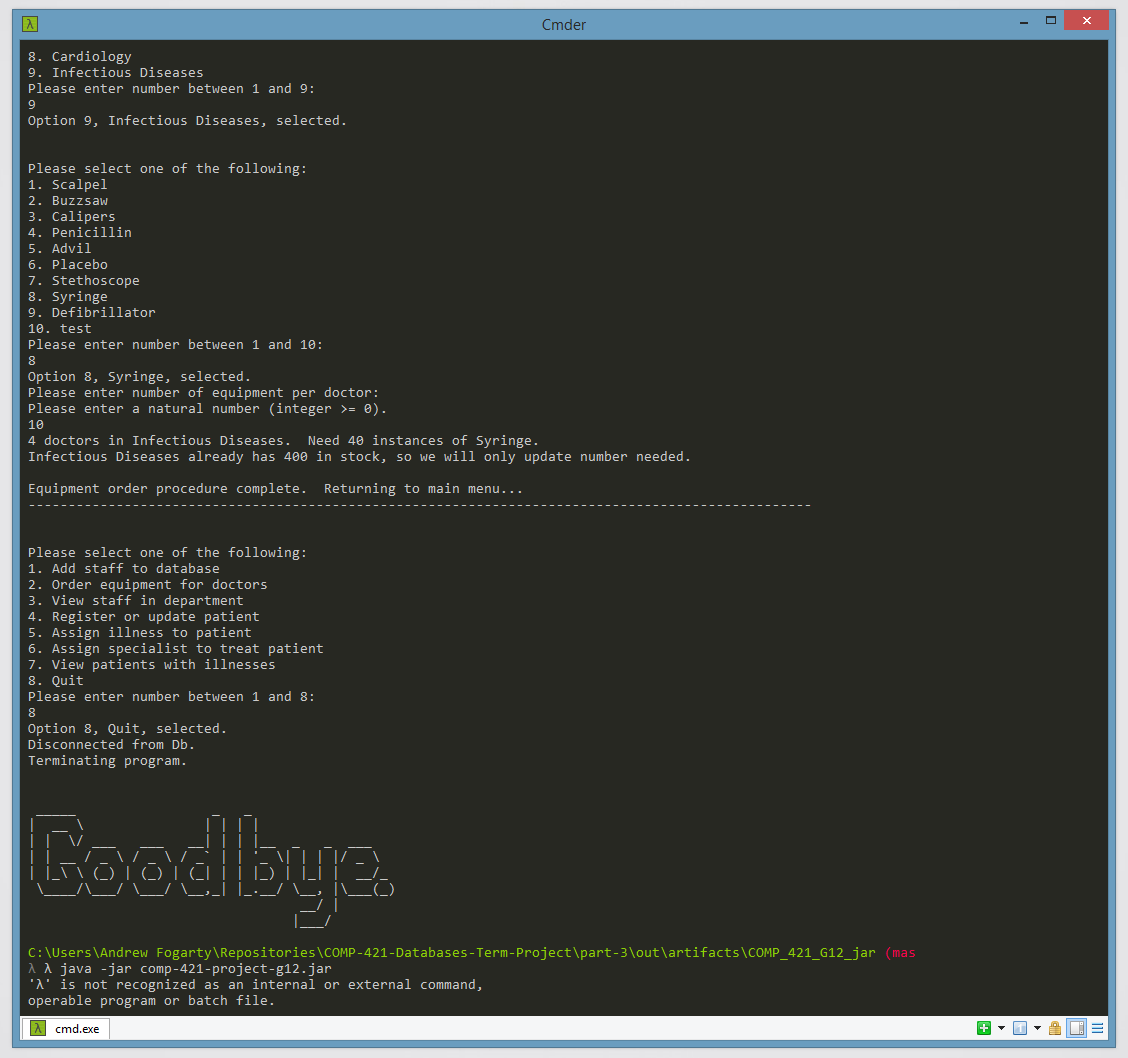
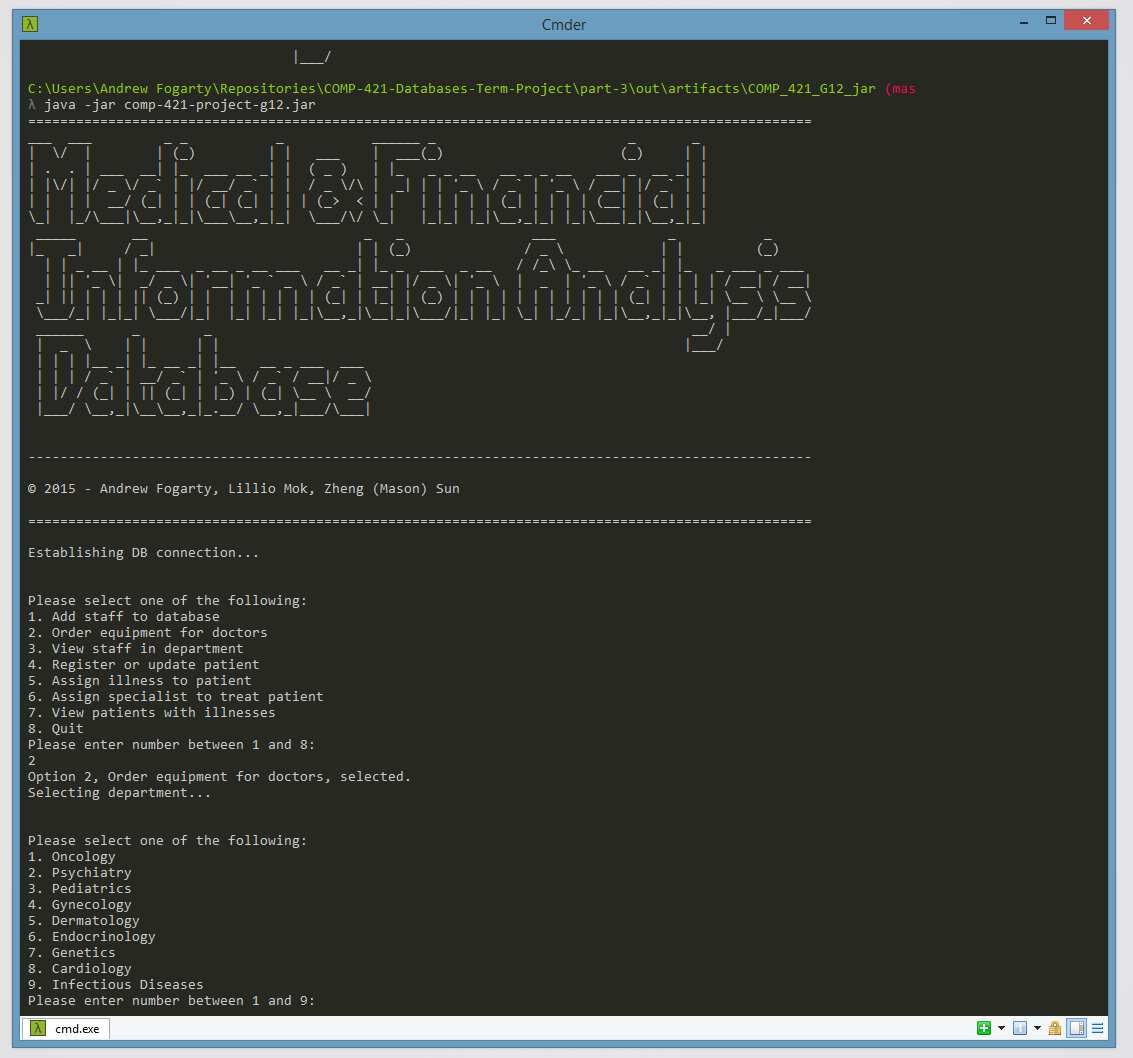
Please see attached for our application and executable. Due to ASCII art being produced on the command line, we cannot fit the output onto one pdf. This is also attached. Below are screenshots of the 7 procedures of our application.

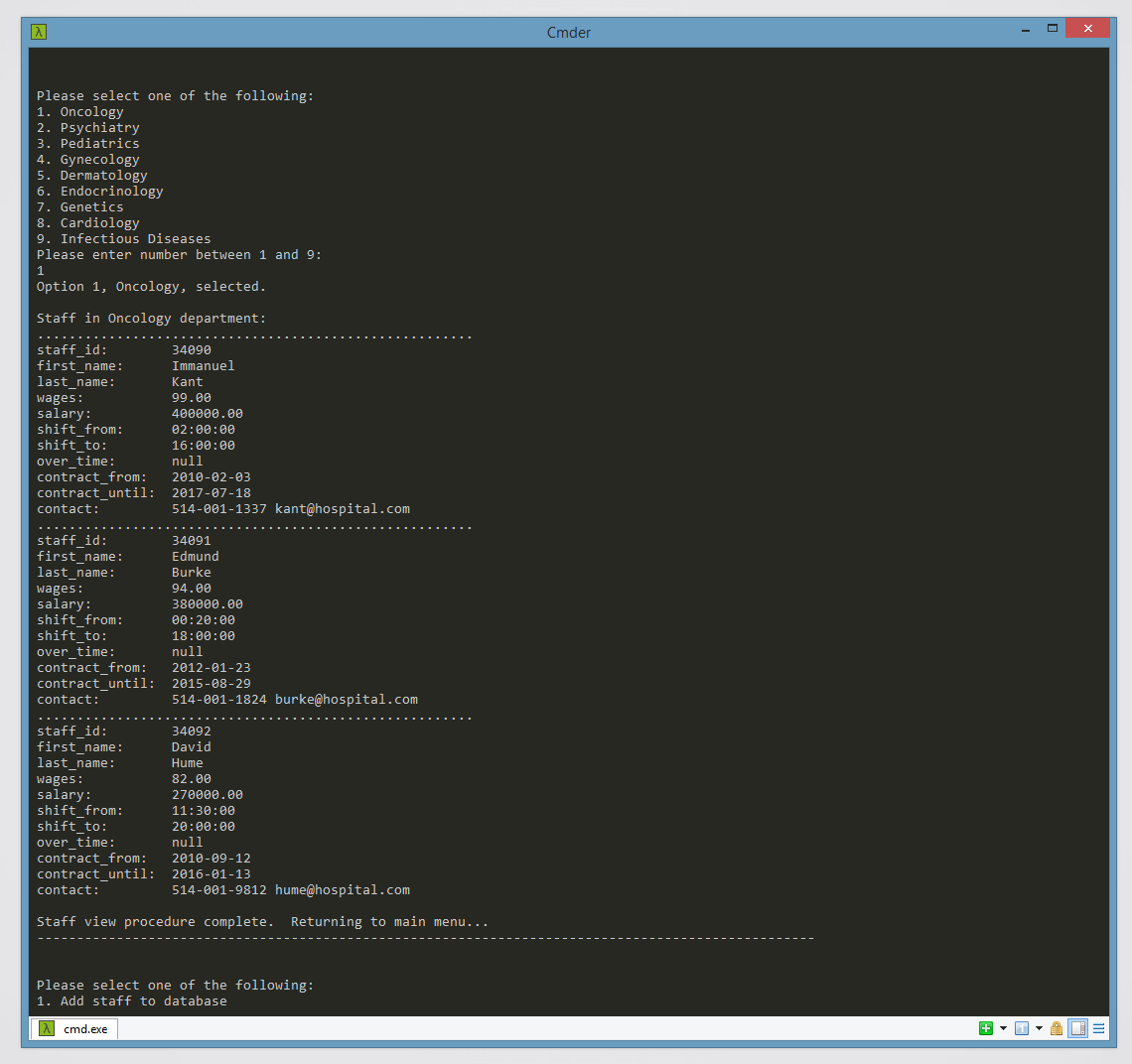
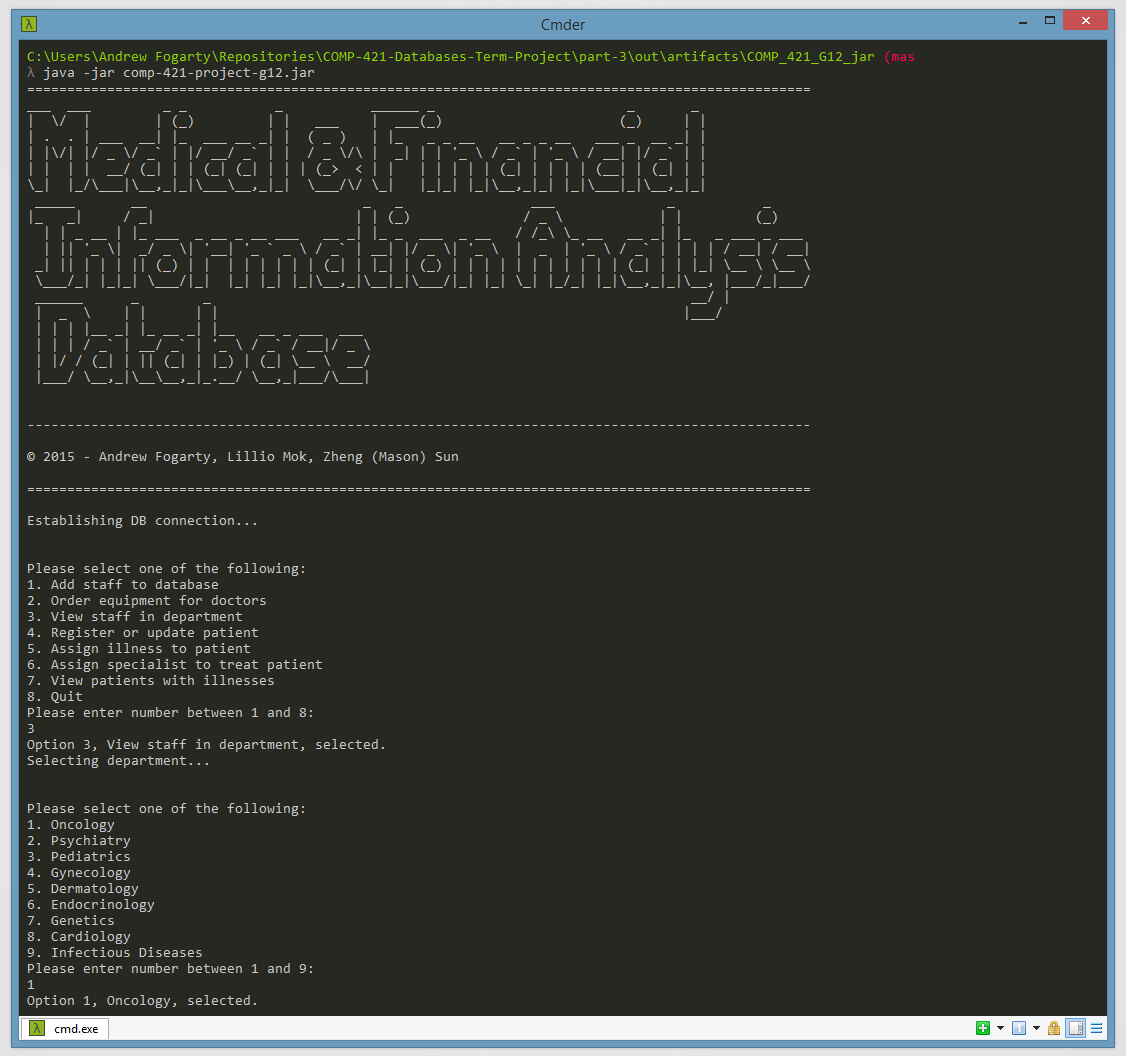
Procedure 1: Insertion into Staff Relation

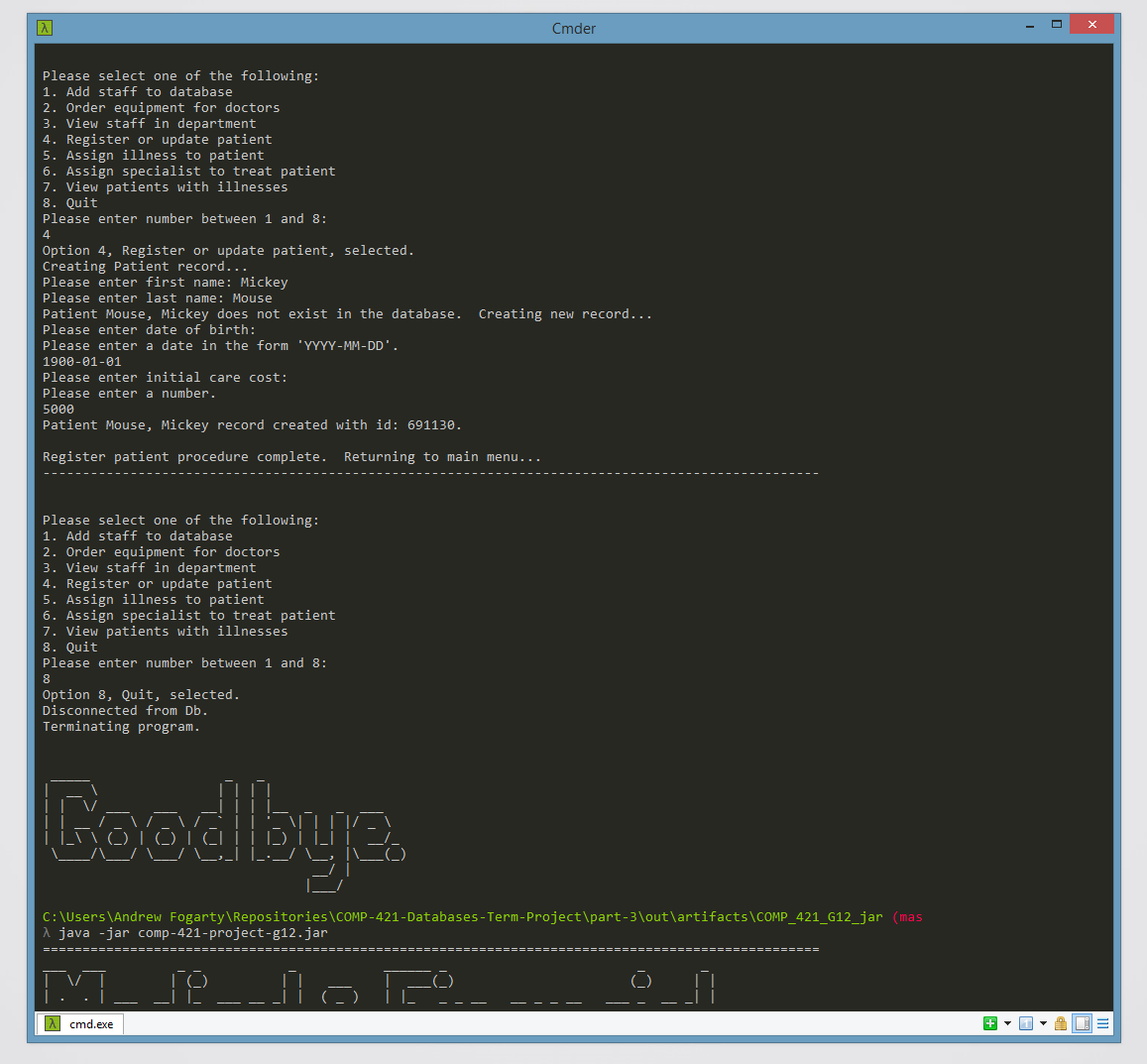
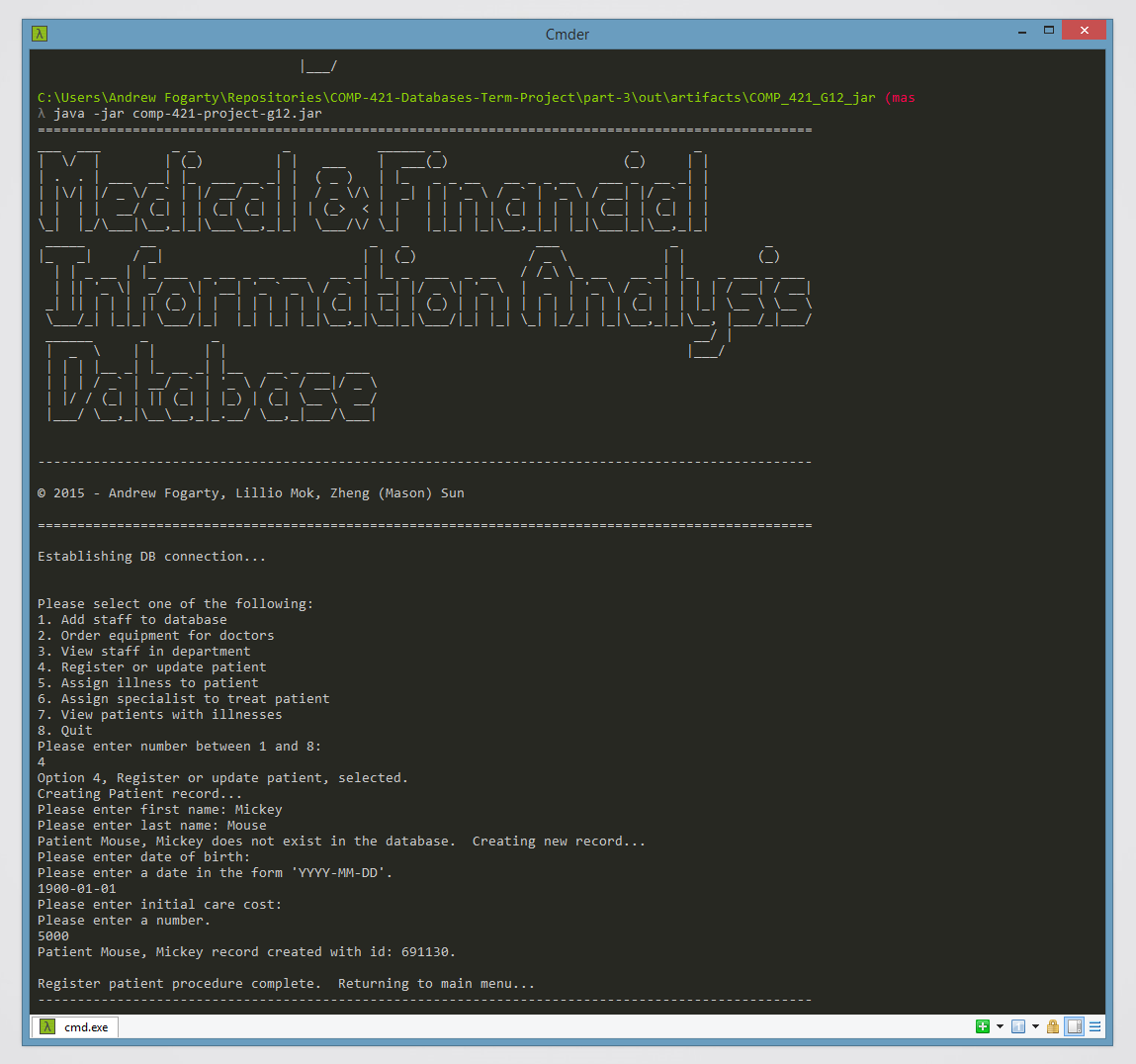


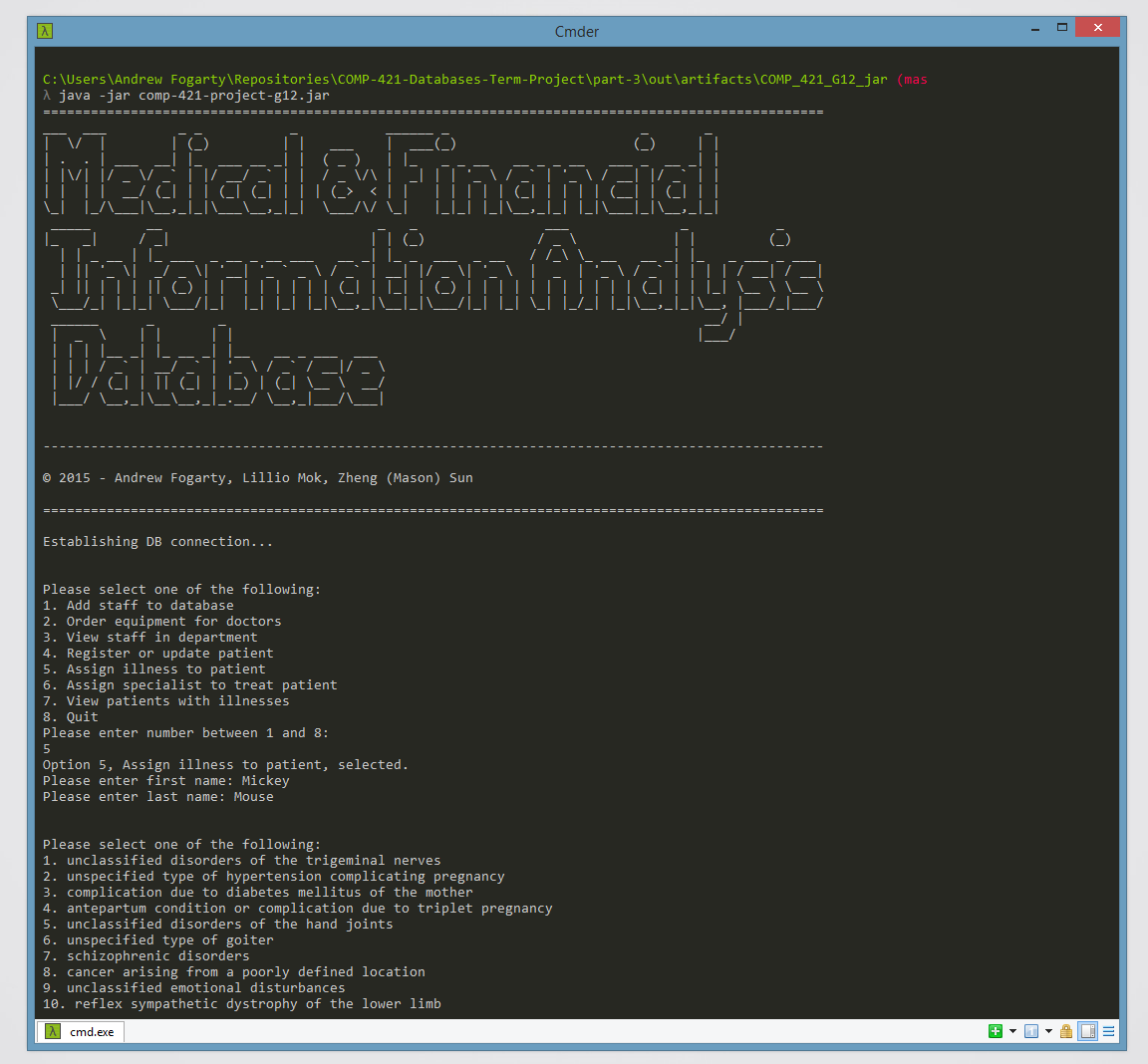




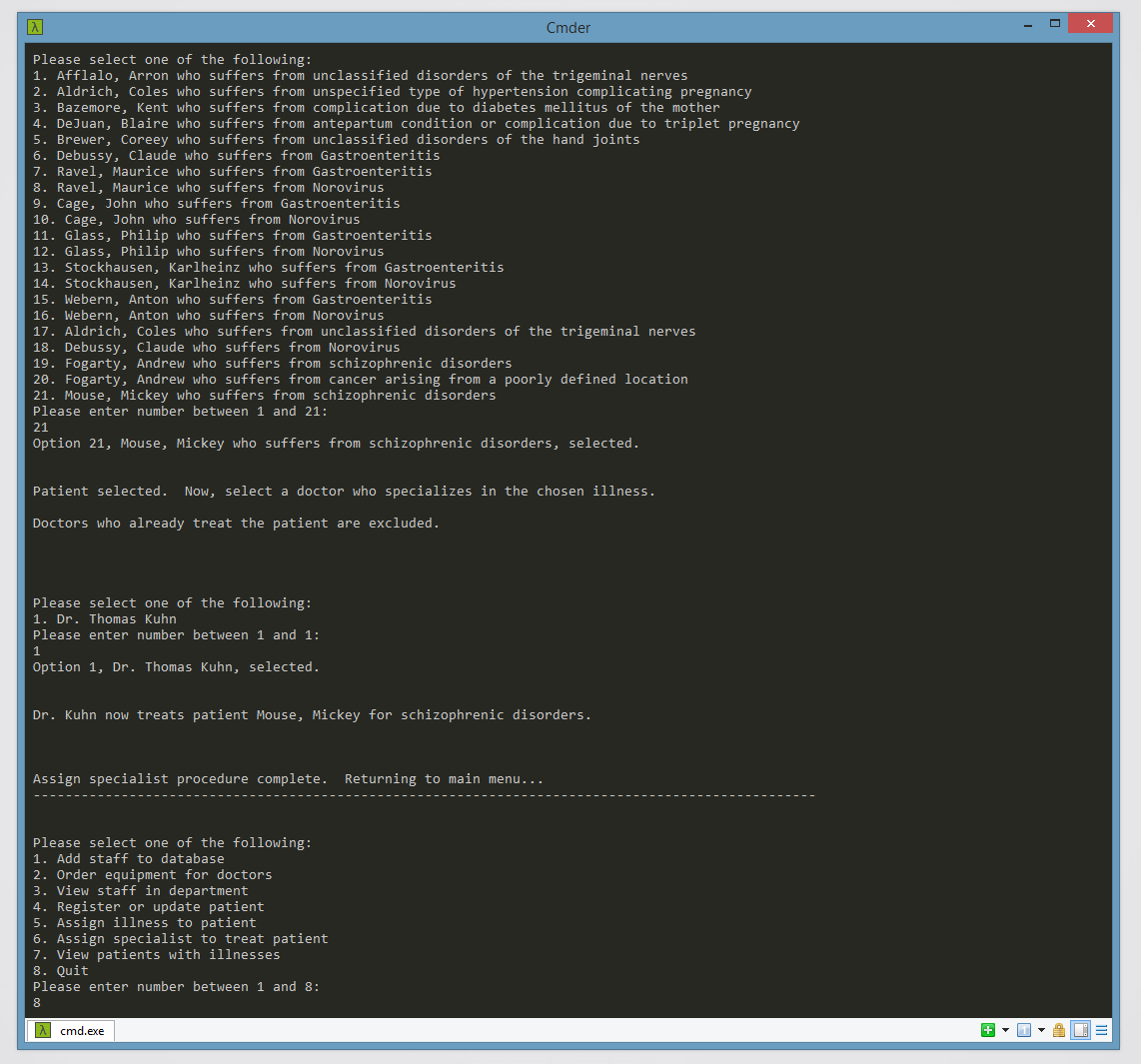
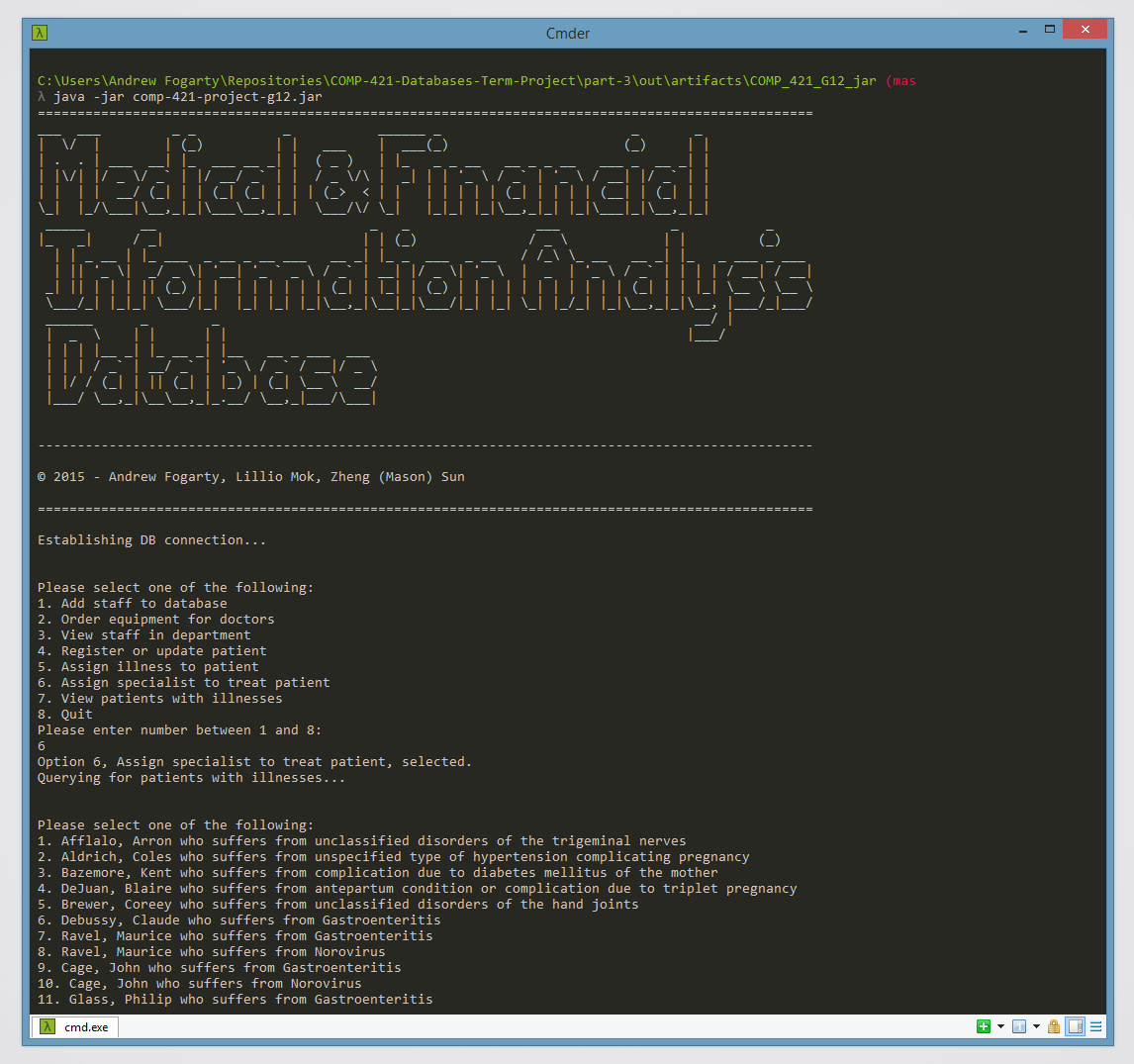
Procedure 2: Order Equipment for Staff who are Doctors

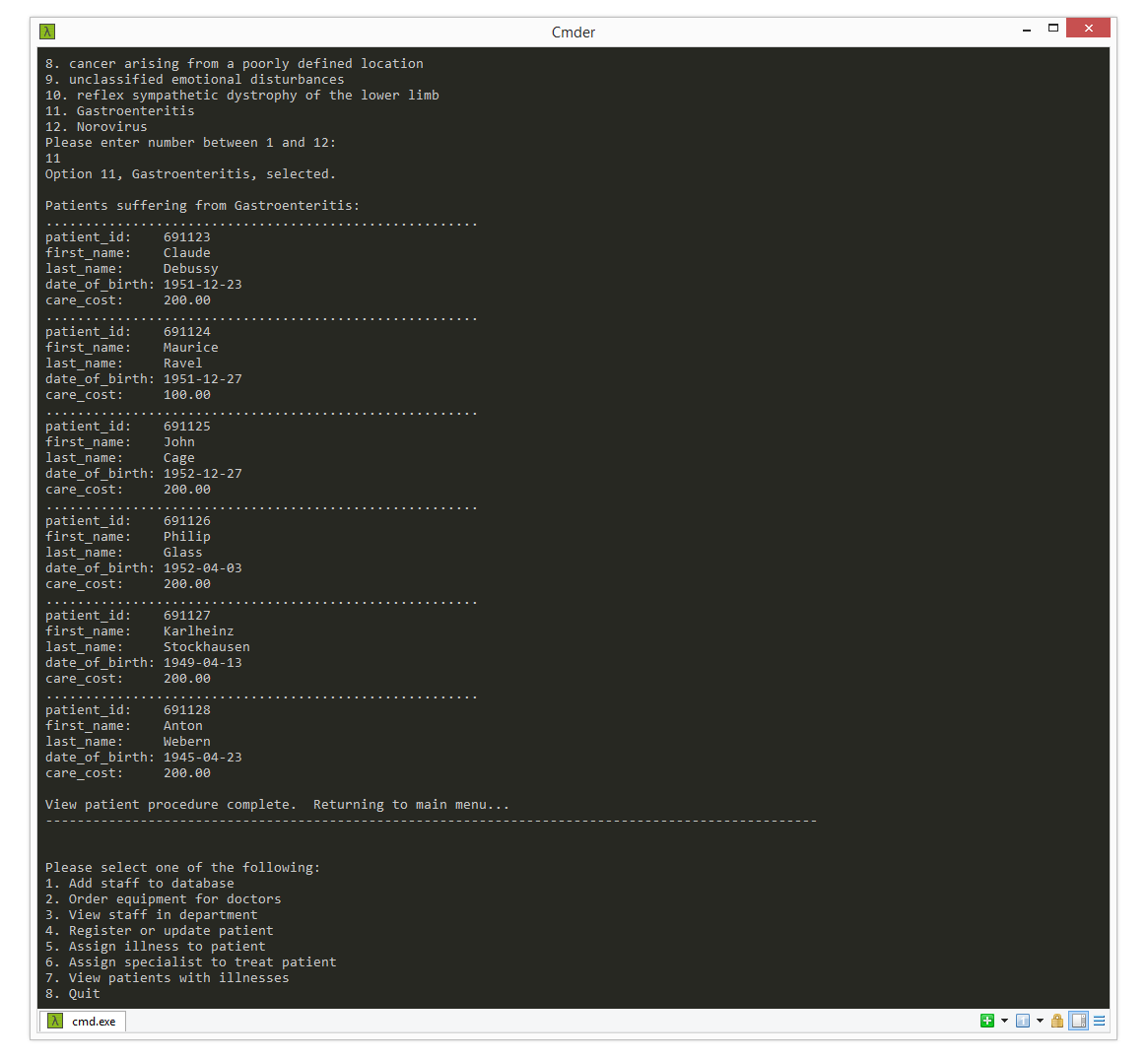
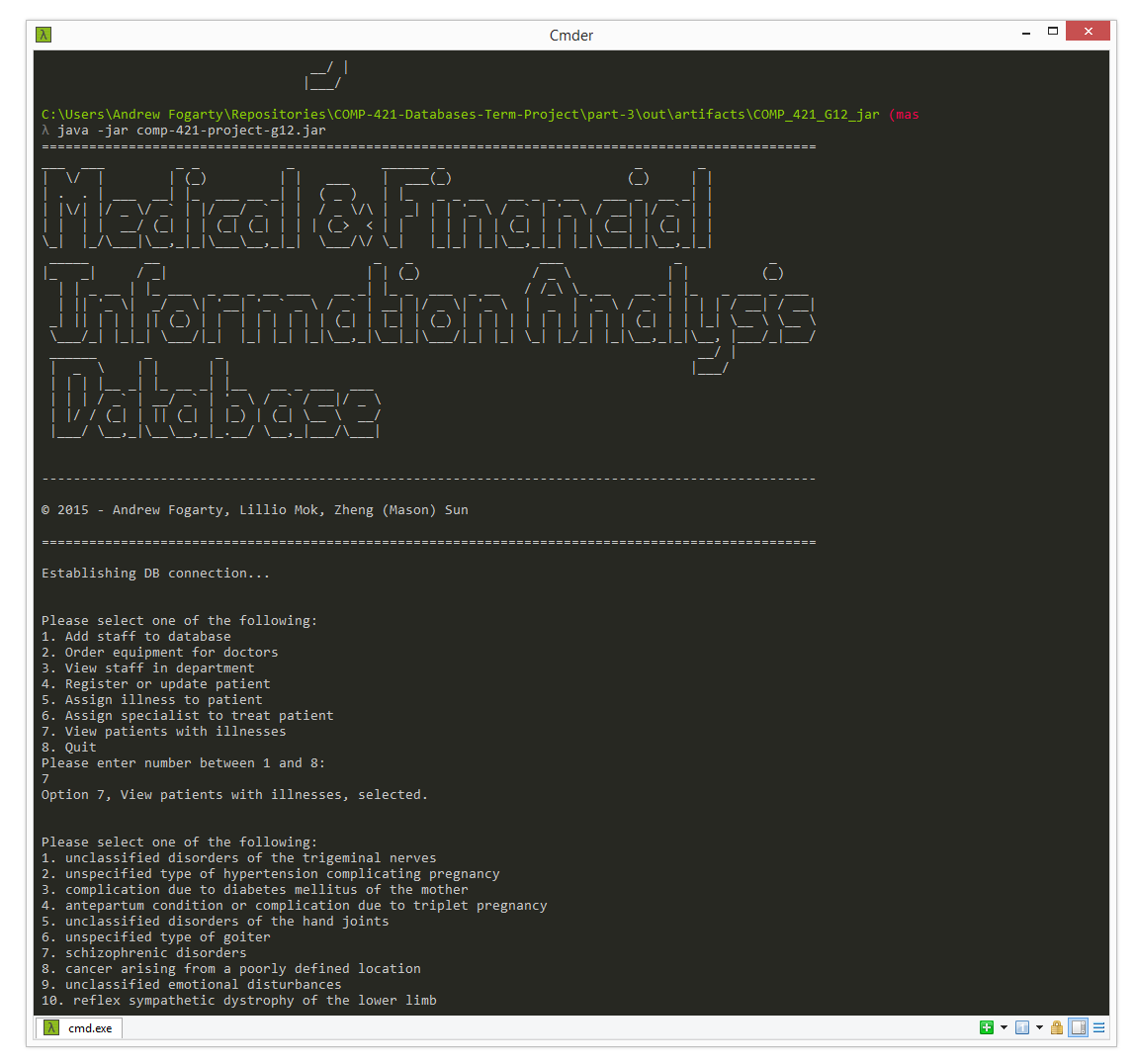
Procedure 3: Search for Staff Members per Department

Procedure 4: Register or Update a Patient

Procedure 5: Assign a Patient to an instance of a Disease



Procedure 6: Assign a Specialist to a Patient for his/her Illness

Procedure 7: View Patients with a specific Illness

The complete scripts of these outputs are found under directory console-print-outs.

Please read README before running our executable.

Source code is under directory source.

**4)**

SELECT S.staff\_id, first\_name, last\_name

FROM Staff S INNER JOIN Doctor D

ON S.staff\_id = D.staff\_id

FULL OUTER JOIN Nurse N

ON S.staff\_id = N.staff\_id

WHERE (

shift\_from <= '03:00:00' -- Shift ends between midnight and 3am

OR shift\_to <='03:00:00' -- Shift begins between midnight and 3am

OR (shift\_to > '03:00:00' AND shift\_from > shift\_to) -- Shift begins before midnight and ends after 3am

);

CREATE INDEX fromInd ON Staff(shift\_from);

CREATE INDEX toInd ON Staff(shift\_to);

CS421=> CREATE INDEX fromInd ON Staff(shift\_from);

CREATE INDEX

CS421=> CREATE INDEX toInd ON Staff(shift\_to);

CREATE INDEX

-- These indices are useful as cost and amount\_needed as they require range searches,

-- and should shorten the time required for the queries. We don’t think that it would make

-- a large amount of difference, however, because of limited records in the relation.

DROP INDEX fromInd;

DROP INDEX toInd;

-- As we guessed, our results with the indexes were around 160ms, while the results without the

-- indexes took around 130ms. This is, again, probably because we don’t have the number of entries

-- to justify the overhead for using indexes, and so there was little difference between the speeds.