CS 7380 DBMS Final Project Report

Lever Group Experimental Data Management System

Group 3

Danlu Liu
Yang Liu
Siman Huang
Peng Zhao

1. Introduction:

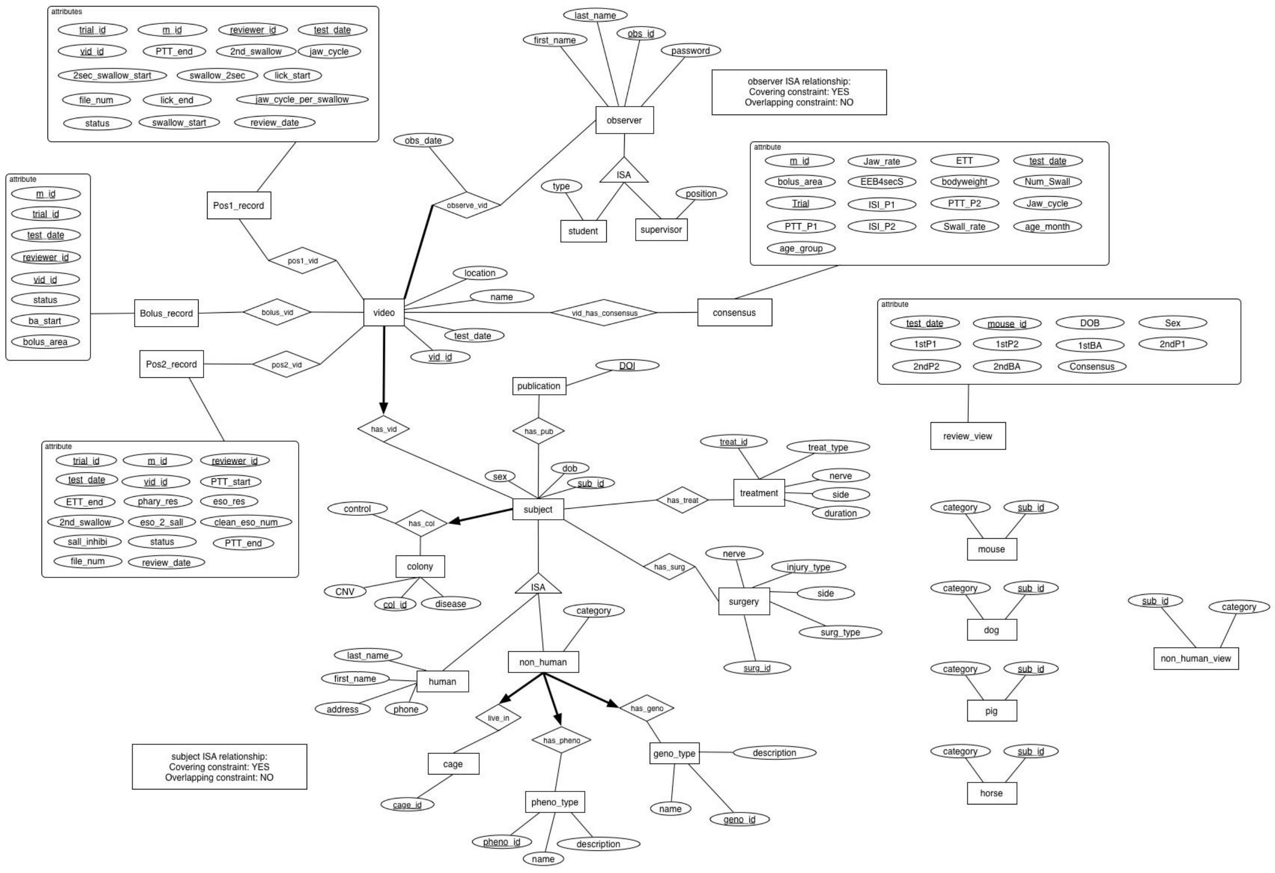
Our database application is used for managing biomedical experimental data with features of raw data collection, data processing and analysis. It's a proprietary application specially designed for our client, but it has the potential to provide subscribers a hassle-free experience of experimental data management.

Our client is Dr. Teresa Lever, an Assistant Professor in Otolaryngology Department in School of Medicine, University of Missouri. Her group investigates dysphagia in amyotrophic lateral sclerosis (ALS) based on different animal models. ALS is a progressive neurodegenerative disease that affects nerve cells in the brain and the spinal cord. The video fluoroscopic swallowing study (VFSS) method they used will result in tons of video records of mice's behaviors under different experiment conditions. Dr. Lever's group needs to manually process and analyze the videos. They use paper forms to record meaningful raw data extracted from the videos. Each video requires two reviewers to record the raw data independently. It also requires a consensus for the two records from a consensus member (e.g, Dr. Lever). If the two records have a discrepancy greater than 10% on a same video, they cannot get consensus and need to be reviewed again until the discrepancy is under 10%. The data with consensus will be stored in excel spreadsheets and then imported to a SPSS database, which has a very simple design with some redundancy issues. The whole process is quite labor intensive. In addition, it's hard to get higher-level information, such as getting their working progress. Dr. Lever hopes to have an efficient experimental data management system.

In this project, we have designed and implemented an experimental data management system for Dr. Lever's group. The management system was developed using PHP, Apache and MySQL database with a user-friendly website interface.

The URL is http://cs4380-group3.centralus.cloudapp.azure.com/lever/index.php
The username and password for administrator are root and root
The username and password for Dr. Lever are Teresa and 1234
The username and password for "Kate" are Kate and 1234
The username and password for "Peng" are Peng and 1234
The username and password for "Siman" are Siman and 1234

2. ERD



There are 37 tables and 2 views in total and they are described below:

“observer” table stores the information of observers for experimental videos. “obs_id” is the primary key and other attributes include “first_name”, “last_name”, “password”. “student” and “supervisor” are sub-tables of “observer” (ISA relationship) to store information of students and supervisors of the project.

“video” table stores the information of experimental videos, such as storage path (“location”), video name (“name”), experiment date (“test_date”) and “vid_id” is the primary key.

“subject” table stores the information of experiment subjects including “sex”, “dob” (date of birth), and “sub_id” is the primary key. “human” and “non_human” are sub-tables of “subject” (ISA relationship). “non_human” may contain more than one type of experiment subject (e.g., mouse, dog, etc) and “category” attribute is used to label the types.

“mouse”, “dog”, “pig”, and “horse” are 4 tables resulted from the horizontal decomposition of “non_human”.

“non_human_view” is a view used to mask the 4 tables “mouse”, “dog”, “pig”, and “horse”.

“cage” contains the ids of all cages the experiment subjects are living in.

“geno_type” and “pheno_type” records all the geno types and pheno types of non-human subjects.

“treatment” and “surgery” contains the information of experiment treatments and surgery for subjects if any.

“consensus” table contains observation records that have been consented. “m_id”, “test_date”, and “Trial” form the composite primary key. Each of them cannot individually identify a record since each mouse can be tested on different experiment days and each experiment has 5 trials.

“Pos1_record”, “Pos2_record”, and “Bolus_record” are 3 tables used to store the raw observation data for position 1, position 2 and bolus area. “m_id”, “test_date”, “trial_id”, and “reviewer_id” form the composite primary key. “reviewer_id” is introduced since each video needs to be analyzed by 2 different observers.

“review_view” is a view used to show the overall study progress. It can indicate whether each video has been analyzed or consented. It serves as an overview of progress.

We have also created tables for relationships between different entities.

The constraints are:

- Total participation between video and observer because each video has at least one observers.
- Total participation between video and subject because each video will have exact one subject.
- Total participation between subject and colony because each subject will have exact one colony.
- Total participation between non_human and cage because each non-human subject will live in only one cage.
- Total participation between non_human and pheno_type because each non-human subject only has one pheno_type.
- Total participation between non_human and geno_type because each non-human subject only has one geno_type.

DDL: 37 tables and 2 views in total

CREATE TABLE observer (obs_id VARCHAR(20), first_name VARCHAR(20), last_name VARCHAR(20), PRIMARY KEY (obs_id));	CREATE TABLE student (obs_id VARCHAR(20), type VARCHAR(20), PRIMARY KEY (obs_id), FOREIGN KEY (obs_id) REFERENCES observer (obs_id) ON DELETE CASCADE);	CREATE TABLE supervisor (obs_id VARCHAR(20), position VARCHAR(20), PRIMARY KEY (obs_id), FOREIGN KEY (obs_id) REFERENCES observer (obs_id) ON DELETE CASCADE);
CREATE TABLE video (	CREATE TABLE surgery (	CREATE TABLE subject (

vid_id VARCHAR(20), test_date DATE, location VARCHAR(100), name VARCHAR(20), PRIMARY KEY (vid_id));	surg_id VARCHAR(20), surg_type VARCHAR(20), PRIMARY KEY (surg_id));	sub_id VARCHAR(20), sex VARCHAR(10), dob DATE, PRIMARY KEY (sub_id));
CREATE TABLE has_vid (vid_id VARCHAR(20), sub_id VARCHAR(20) NOT NULL, PRIMARY KEY (vid_id), FOREIGN KEY (vid_id) REFERENCES video (vid_id), FOREIGN KEY (sub_id) REFERENCES subject (sub_id) ON DELETE NO ACTION);	CREATE TABLE has_treat (treat_id VARCHAR(20), sub_id VARCHAR(20), PRIMARY KEY (sub_id, treat_id), FOREIGN KEY (sub_id) REFERENCES subject (sub_id), FOREIGN KEY (treat_id) REFERENCES treatment (treat_id));	CREATE TABLE has_pub (sub_id VARCHAR(20), doi VARCHAR(30), PRIMARY KEY (sub_id, doi), FOREIGN KEY (sub_id) REFERENCES subject (sub_id), FOREIGN KEY (doi) REFERENCES publication (doi));
CREATE TABLE treatment (treat_id VARCHAR(20), treat_type VARCHAR(20), PRIMARY KEY (treat_id));	CREATE TABLE publication (doi VARCHAR(30), PRIMARY KEY(doi));	CREATE TABLE colony (col_id VARCHAR(20), disease VARCHAR(20), CNV VARCHAR(20), PRIMARY KEY (col_id));
CREATE TABLE has_col (col_id VARCHAR(20) NOT NULL, sub_id VARCHAR(20), control BOOLEAN, PRIMARY KEY (sub_id), FOREIGN KEY (sub_id) REFERENCES subject (sub_id), FOREIGN KEY (col_id) REFERENCES colony (col_id) ON DELETE NO ACTION);	CREATE TABLE observe_vid (obs_id VARCHAR(20) NOT NULL, vid_id VARCHAR(20), obs_date DATE, PRIMARY KEY (vid_id), FOREIGN KEY (obs_id) REFERENCES observer (obs_id), FOREIGN KEY (vid_id) REFERENCES video (vid_id));	CREATE TABLE has_surg (surg_id VARCHAR(20), sub_id VARCHAR(20), PRIMARY KEY (surg_id, sub_id), FOREIGN KEY (sub_id) REFERENCES subject (sub_id), FOREIGN KEY (surg_id) REFERENCES surgery (surg_id));
CREATE TABLE human (sub_id VARCHAR(20), first_name VARCHAR(20), last_name VARCHAR(20), address VARCHAR(100), phone VARCHAR(20), PRIMARY KEY (sub_id), FOREIGN KEY (sub_id) REFERENCES subject (sub_id) ON DELETE CASCADE);	CREATE TABLE non_human (sub_id VARCHAR(20), category VARCHAR(20), PRIMARY KEY (sub_id), FOREIGN KEY (sub_id) REFERENCES subject (sub_id) ON DELETE CASCADE);	CREATE TABLE has_pheno (sub_id VARCHAR(20), pheno_id VARCHAR(20) NOT NULL, PRIMARY KEY (sub_id), FOREIGN KEY (sub_id) REFERENCES non_human (sub_id), FOREIGN KEY (pheno_id) REFERENCES pheno_type (pheno_id) ON DELETE NO ACTION);

CREATE TABLE pig(sub_id VARCHAR(20), category VARCHAR(20), PRIMARY KEY (sub_id));	CREATE TABLE horse(sub_id VARCHAR(20), category VARCHAR(20), PRIMARY KEY (sub_id)); CREATE TABLE dog(sub_id VARCHAR(20), category VARCHAR(20), PRIMARY KEY (sub_id));	CREATE TABLE dog(sub_id VARCHAR(20), category VARCHAR(20), PRIMARY KEY (sub_id));
CREATE TABLE geno_type (geno_id VARCHAR(20), name VARCHAR(20), PRIMARY KEY (geno_id));	CREATE TABLE pheno_type (pheno_id VARCHAR(20), name VARCHAR(20), PRIMARY KEY (pheno_id));	CREATE TABLE cage (cage_id VARCHAR(20), PRIMARY KEY (cage_id));
CREATE TABLE live_in (sub_id VARCHAR(20), cage_id VARCHAR(20) NOT NULL, PRIMARY KEY (sub_id), FOREIGN KEY (sub_id) REFERENCES non_human (sub_id), FOREIGN KEY (cage_id) REFERENCES cage (cage_id) ON DELETE NO ACTION);	CREATE TABLE has_geno (sub_id VARCHAR(20), geno_id VARCHAR(20) NOT NULL, PRIMARY KEY (sub_id), FOREIGN KEY (sub_id) REFERENCES non_human (sub_id), FOREIGN KEY (geno_id) REFERENCES geno_type (geno_id) ON DELETE NO ACTION);	CREATE TABLE Bolus_record (trial_id INTEGER, m_id VARCHAR(20), reviewer_id VARCHAR(20), test_date DATE, status VARCHAR(20), vid_id VARCHAR(20), PRIMARY KEY (m_id,trial_id,reviewer_id,test_date,vid_id));
CREATE TABLE vid_has_consensus (m_id VARCHAR(20), vid_id VARCHAR(20), PRIMARY KEY (m_id, vid_id), FOREIGN KEY (m_id) REFERENCES consensus (m_id), FOREIGN KEY (vid_id) REFERENCES video (vid_id));	CREATE TABLE Raw_trial_data (trial_id INTEGER, status VARCHAR(20), PRIMARY KEY (trial_id));	CREATE TABLE con_has_raw (m_id VARCHAR(20) NOT NULL, trial_id INTEGER NOT NULL, PRIMARY KEY (trial_id, m_id), FOREIGN KEY (trial_id) REFERENCES Raw_trial_data (trial_id), FOREIGN KEY (m_id) REFERENCES consensus (m_id) ON DELETE NO ACTION);
CREATE TABLE Pos1_record (trial_id INTEGER, m_id VARCHAR(20), reviewer_id VARCHAR(20), test_date DATE,	CREATE TABLE Pos2_record (trial_id INTEGER, m_id VARCHAR(20), reviewer_id VARCHAR(20), test_date DATE,	CREATE TABLE consensus (m_id VARCHAR(20), Jaw_rate REAL, ETT REAL, test_date DATE,

```
swallow_start VARCHAR(40),
PTT_end VARCHAR(40),
phary_res REAL,
2nd_swallow VARCHAR(40),
tongue_cycle_per_swallow
REAL,
jaw_cycle REAL,
2sec_swallow REAL,
swallow_2sec REAL,
lick_start VARCHAR(40),
file_num VARCHAR(10),
lick_end VARCHAR(40),
tongue_cycle REAL,
jaw_cycle_per_swallow REAL,
status VARCHAR(20),
vid_id VARCHAR(20),
PRIMARY KEY
(m_id,trial_id,reviewer_id,test_d
ate,vid_id)
);
```

```
PTT_start VARCHAR(40),
PTT_end VARCHAR(40),
ETT_end VARCHAR(40),
phary_res REAL,
eso_res REAL,
2nd_swallow VARCHAR(40),
eso_2_sall VARCHAR(40),
clean_eso_num REAL,
sall_inhibi REAL,
status VARCHAR(20),
vid_id VARCHAR(20),
PRIMARY KEY
(m_id,trial_id,reviewer_id,test_d
ate,vid_id)
);
```

```
bolus_area REAL,
EEB4secS REAL,
bodyweight REAL,
Num_Swall REAL,
Trial REAL,
ISI_P1 REAL,
PTT_P2 REAL,
Jaw_cycle REAL,
PTT_P1 REAL,
ISI_P2 REAL,
Swall_rate REAL,
age_month REAL,
age_group VARCHAR(20),
PRIMARY KEY
(m_id,test_date,Trial)
);
```

```
CREATE TABLE has_p1_record
(
vid_id_p1 VARCHAR(20),
trial_id INTEGER,
m_id VARCHAR(20),
reviewer_id VARCHAR(20),
test_date DATE,
vid_id_vi VARCHAR(20),
PRIMARY KEY
(m_id,trial_id,reviewer_id,test_d
ate,vid_id_p1,vid_id_vi),
FOREIGN KEY (vid_id_vi)
REFERENCES mouse_db.video
(vid_id),
FOREIGN KEY (vid_id_p1)
REFERENCES
mouse_db.Pos1_record (vid_id)
);
```

```
CREATE TABLE has_p2_record
(
vid_id_p2 VARCHAR(20),
trial_id INTEGER,
m_id VARCHAR(20),
reviewer_id VARCHAR(20),
test_date DATE,
vid_id_vi VARCHAR(20),
PRIMARY KEY
(m_id,trial_id,reviewer_id,test_d
ate,vid_id_p2,vid_id_vi),
FOREIGN KEY (vid_id_vi)
REFERENCES mousedb.video
(vid_id),
FOREIGN KEY (vid_id_p2)
REFERENCES
mousedb.Pos2_record (vid_id)
);
```

```
CREATE TABLE has_bolus (
trial_id INTEGER,
m_id VARCHAR(20),
reviewer_id VARCHAR(20),
test_date DATE,
vid_id_bo VARCHAR(20),
vid_id_vi VARCHAR(20),
PRIMARY KEY
(m_id,trial_id,reviewer_id,test_d
ate,vid_id_bo,vid_id_vi),
FOREIGN KEY (vid_id_vi)
REFERENCES mousedb.video
(vid_id),
FOREIGN KEY (vid_id_bo)
REFERENCES
mousedb.Bolus_record (vid_id)
);
```

```
CREATE VIEW review_view AS
SELECT DISTINCT V.test_date, S.sub_id, S.dob, S.sex,
        IF(S.sub_id IN (SELECT P1.m_id FROM Pos1_record P1 WHERE P1.test_date = V.test_date AND
P1.m_id = S.sub_id AND P1.status = 'r1'), (SELECT P1.reviewer_id FROM Pos1_record P1 WHERE
P1.test_date = V.test_date AND P1.m_id = S.sub_id AND P1.status = 'r1'), ' ') AS 1stP1,
        IF(S.sub_id IN (SELECT P1.m_id FROM Pos1_record P1 WHERE P1.test_date = V.test_date AND
P1.m_id = S.sub_id AND P1.status = 'r2'), (SELECT P1.reviewer_id FROM Pos1_record P1 WHERE
P1.test_date = V.test_date AND P1.m_id = S.sub_id AND P1.status = 'r2'), ' ') AS 2ndP1,
        IF(S.sub_id IN (SELECT P2.m_id FROM Pos2_record P2 WHERE P2.test_date = V.test_date AND
P2.m_id = S.sub_id AND P2.status = 'r1'), (SELECT P2.reviewer_id FROM Pos2_record P2 WHERE
P2.test_date = V.test_date AND P2.m_id = S.sub_id AND P2.status = 'r1'), ' ') AS 1stP2,
        IF(S.sub_id IN (SELECT P2.m_id FROM Pos2_record P2 WHERE P2.test_date = V.test_date AND
P2.m_id = S.sub_id AND P2.status = 'r2'), (SELECT P2.reviewer_id FROM Pos2_record P2 WHERE
P2.test_date = V.test_date AND P2.m_id = S.sub_id AND P2.status = 'r2'), ' ') AS 2ndP2,
        IF(S.sub_id IN (SELECT B.m_id FROM Bolus_record B WHERE B.test_date = V.test_date AND
B.m_id = S.sub_id AND B.status = 'r1'), (SELECT B.reviewer_id FROM Bolus_record B WHERE
B.test_date = V.test_date AND B.m_id = S.sub_id AND B.status = 'r1'), ' ') AS 1stBA,
        IF(S.sub_id IN (SELECT B.m_id FROM Bolus_record B WHERE B.test_date = V.test_date AND
B.m_id = S.sub_id AND B.status = 'r2'), (SELECT B.reviewer_id FROM Bolus_record B WHERE
B.test_date = V.test_date AND B.m_id = S.sub_id AND B.status = 'r2'), ' ') AS 2ndBA,
        IF(S.sub_id IN (SELECT P1.m_id FROM Pos1_record P1, Pos2_record P2 WHERE P2.m_id =
P1.m_id AND P1.m_id = S.sub_id AND P1.test_date = V.test_date AND P2.test_date = V.test_date AND
P1.status = 'c' AND P2.status = 'c'), 'YES', 'NO') AS consensus
FROM video V, Pos1_record P1, Pos2_record P2, Bolus_record B, subject S, has_vid H
WHERE V.vid_id = H.vid_id AND H.sub_id = S.sub_id;
```

```
CREATE TABLE mouse LIKE non_human_test;
INSERT INTO mouse SELECT * FROM non_human_test where category = 'Mouse';
CREATE TABLE dog LIKE non_human_test;
INSERT INTO dog SELECT * FROM non_human_test where category = 'Dog';
CREATE TABLE pig LIKE non_human_test;
INSERT INTO pig SELECT * FROM non_human_test where category = 'Pig';
CREATE TABLE horse LIKE non_human_test;
INSERT INTO horse SELECT * FROM non_human_test where category = 'Horse';
CREATE VIEW non_human_view AS
SELECT * FROM mouse
UNION
SELECT * FROM dog
UNION
SELECT * FROM pig
UNION
SELECT * FROM horse;
```

3. Queries

3.1. Find the mouse IDs whose name contains the letter “H”, which is used when users want to do fuzzy surch.

```
SELECT sub_id
FROM mouse
WHERE sub_id LIKE "%H%";
```

3.2. Find the number of missing value of jaw rate from consensuses data of each mouse which helps to know which mouse need to do another experiment.

```
SELECT DISTINCT(m_id), count(*)
FROM consensus
WHERE Jaw_rate
IS NULL group by (m_id);
```

3.3. Find out average value of observed behavior among five trials, our client on a specific test date.

```
SELECT m_id,test_date,avg(ISI_P2) AS isi2,avg(PTT_P2) AS ptt2,
count(EEB4secS) AS ee
FROM consensus
WHERE test_date = '2014-08-08'
GROUP BY m_id,test_date
```

3.4. Find out which mouse has 2 kinds of phenotype.

```
SELECT DISTINCT sub_id, count(*) as num
FROM has_pheno
HAVING count(*) = 2;
```

3.5. Find the number of experiments students has reviewed of position 1.

```
SELECT first_name, count(*)
FROM Pos1_record,observer
WHERE observer.obs_id=Pos1_record.reviewer_id GROUP BY reviewer_id;
```

3.6. Find out the oldest and youngest mouse that has been tested?

Oldest:

```
SELECT distinct(S.sub_id)
FROM consensus C, subject S
WHERE S.sub_id=C.m_id
ORDER abs(DATEDIFF(C.test_date, S.dob))  DESC limit 1;
```

Youngest:

```
SELECT distinct(S.sub_id)
FROM consensus C, subject S
WHERE S.sub_id=C.m_id
ORDER abs(DATEDIFF(C.test_date, S.dob))  ASC limit 1;
```

3.7. Find out how many mice didn’t take any treatment, which helps to know how many mice need treatment.

```
SELECT count(S.sub_id)
FROM subject S
WHERE S.sub_id NOT IN
(SELECT HT.sub_id FROM has_treat HT);
```

3.8. Find out all mouse id that has not been published, which helps to focus on new mouse study.

```
SELECT S.sub_id
FROM subject S
WHERE S.sub_id NOT IN
(SELECT HP.sub_id FROM has_pub HP);
```

3.9. Find out who observed mouse ‘L124’ on which day, which helps to double check question observed record.

```
SELECT O.first_name,O.last_name,P.review_date
FROM observer O, Pos1_record P
```

```
WHERE P.m_id= 'H601' AND P.reviewer_id= O.obs_id;
```

3.10. Find out the phenotype and genotype of mouse ‘L124’.

```
SELECT PH.name, HPH.sub_id,G.name
FROM geno_type G,pheno_type PH ,has_pheno HPH, has_geno HG
WHERE PH.pheno_id = HPH.pheno_id AND HG.sub_id= HPH.sub_id AND
HG.geno_id=G.geno_id AND HG.sub_id = 'L124';
```

3.11. Find out mouse that has not taken videos, which mean has not been exposed to X-rays, which helps to see the effect of X-rays on moues.

```
SELECT S.sub_id
FROM subject S
WHERE S.sub_id NOT IN
( SELECT m_id FROM vid_has_consensus);
```

3.12. Find out the number of reviewed mouse at ‘End-stage’ age group by each colony, which helps to see the distribution of almost-died mouse among colonies.

```
SELECT CO.disease,count(*)
FROM consensus C, has_col HC,colony CO
WHERE C.age_group = 'End-stage' and HC.sub_id=C.m_id and
HC.col_id=CO.col_id
GROUP BY CO.disease;
```

3.13. Find the number of mouse taken all surgery.

```
SELECT S.sub_id
FROM subject S
WHERE NOT EXISTS
(SELECT B.surg_id
FROM surgery B
WHERE B.surg_id NOT IN
 (SELECT B.surg_id
FROM has_surg R
WHERE R.sub_id=S.sub_id));
```

3.14. Find the number of records of mouse ‘H106’ that is taken after ‘2014-08-08’.

```
SELECT distinct(vid_id), count(*)
FROM Pos1_record  where review_date > '2000-01-01' AND m_id='H601' GROUP
BY vid_id;
```

3.15. Find out mouse id, age_group,body weight has swallow rate between 3 and 5, which helps to detect relation among swallow behavior, age and body weight.

```
SELECT distinct(m_id),age_group,body_weight
FROM consensus
WHERE  Swall_rate >= 3 AND Swall_rate <= 5;
```

3.16. Find out mouse id which is at ‘End-stage’ age group but their actual age is less than 7 month, which helps to detect unexpected death.

```
SELECT distinct(m_id)
FROM consensus
WHERE age_group = 'End-stage' AND age_month < 7 ;
```

3.17. Find out the number of times exposed to x-rays of all mouse that has taken videos since videos were taken under x-ray environment.

```
SELECT m_id,count(*)
FROM vid_has_consensus
GROUP BY m_id;
```

3.18. Find if the login information is correct.

```
SELECT first_name, password
FROM observer
WHERE first_name='Teresa' AND password='1234';
```

3.19. Find the student who hasn't review a record yet

```
SELECT obs_id
FROM observer
WHERE obs_id NOT IN (select reviewer_id from Pos1_record);
```

3.20. Find the user who is neither a supervisor or a student.

```
SELECT first_name, last_name
FROM observer O where O.obs_id
NOT IN
(SELECT obs_id
FROM student
UNION select obs_id
FROM supervisor);
```

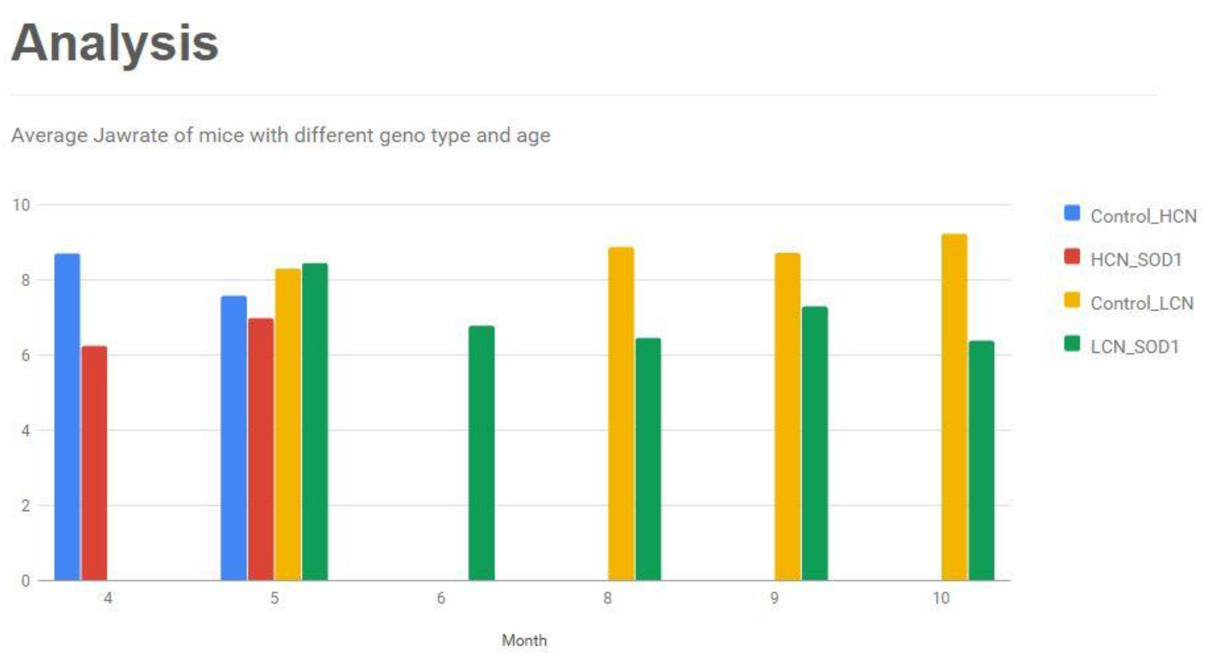
4. Analytics

Our clients do the research on Mice for different diseases. To help them have an overall view of its experiments, we need to do the following analysis. In our project, we use pie charts, bar charts, line charts and tables (which can sort for each column) to show the analysis.

In this part, I use some sql sentences contains:

- **ABS, DATEDIFF, COUNT(*), AVG, ROUND, MONTH, YEAR, is NULL.**
- **NOT IN, LEFT JOIN, NATURAL JOIN, GROUP BY, ORDER BY, DISTINCT.**
- **Nested Query, 'AS' sentences**

1. Jaw rate comparison between different age group and colony group.



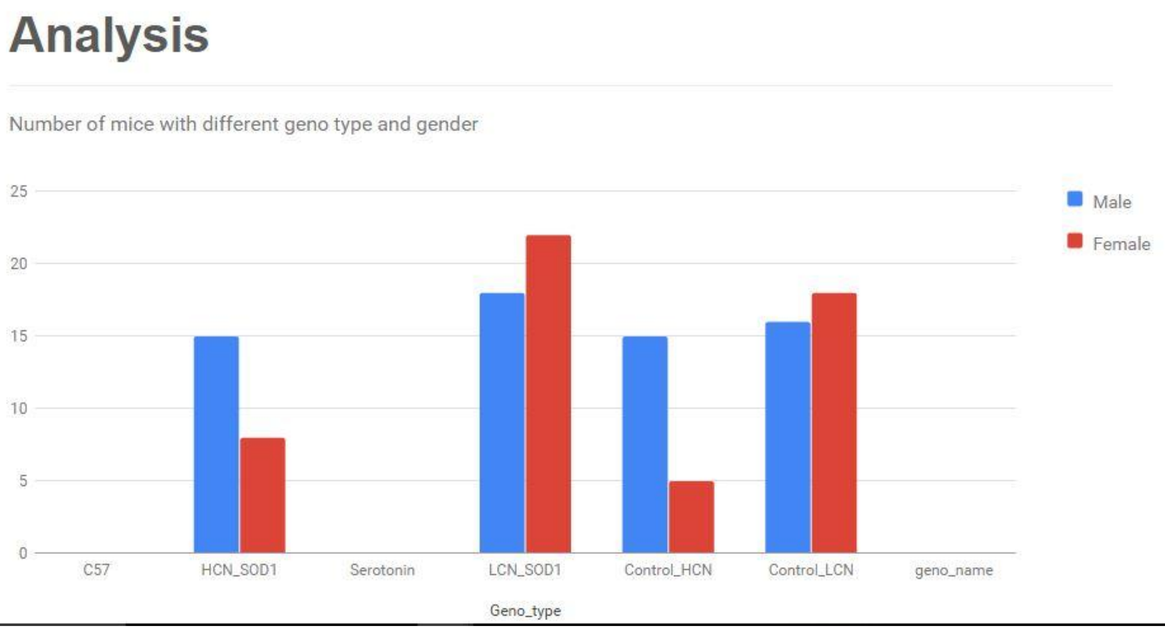
SQL:

//List average Jaw rate with different geno type and age(month) for tested mice from consensus //experiment data
select age,name,avg(rate) from (select m_id,avg(Jaw_rate) as rate,test_date,round(age_month,0) as age,name from consensus join (select sub_id,name from geno_type join (select * from subject natural join non_human_view natural join has_geno) as test on (geno_type.geno_id=test.geno_id))as geno on (geno.sub_id=consensus.m_id) group by m_id, age_month) as s group by s.age,s.name;

//List geno type of tested mice(from consensus table)
select distinct(s2.name) from (select age,name,avg(rate) from (select m_id,avg(Jaw_rate) as rate,test_date,round(age_month,0) as age,name from consensus join (select sub_id,name from geno_type join (select * from subject natural join non_human_view natural join has_geno) as test on (geno_type.geno_id=test.geno_id))as geno on (geno.sub_id=consensus.m_id) group by m_id, age_month) as s group by s.age,s.name) as s2;

//List different mice age intervals from tested mice(from consensus table)
select distinct(age) from (select age,name,avg(rate) from (select m_id,avg(Jaw_rate) as rate,test_date,round(age_month,0) as age,name from consensus join (select sub_id,name from geno_type join (select * from subject natural join non_human_view natural join has_geno) as test on (geno_type.geno_id=test.geno_id))as geno on (geno.sub_id=consensus.m_id) group by m_id, age_month) as s group by s.age,s.name) as s2;

2. Number of mice in different gender by genotype.



SQL:
//List Mice number for different geno type and different gender
select sex,geno_type.geno_id,geno_type.name,count(*) from geno_type left join (select * from subject natural join non_human_view natural join has_geno) as test on (test.geno_id=geno_type.geno_id) group by geno_type.geno_id,sex;

//List the geno types' names
select geno_type.name from geno_type;

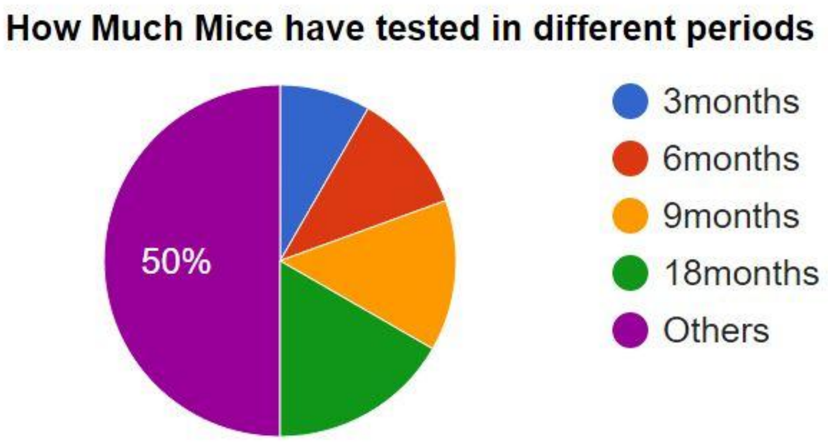
3. The distribution of mice tested in different time period.

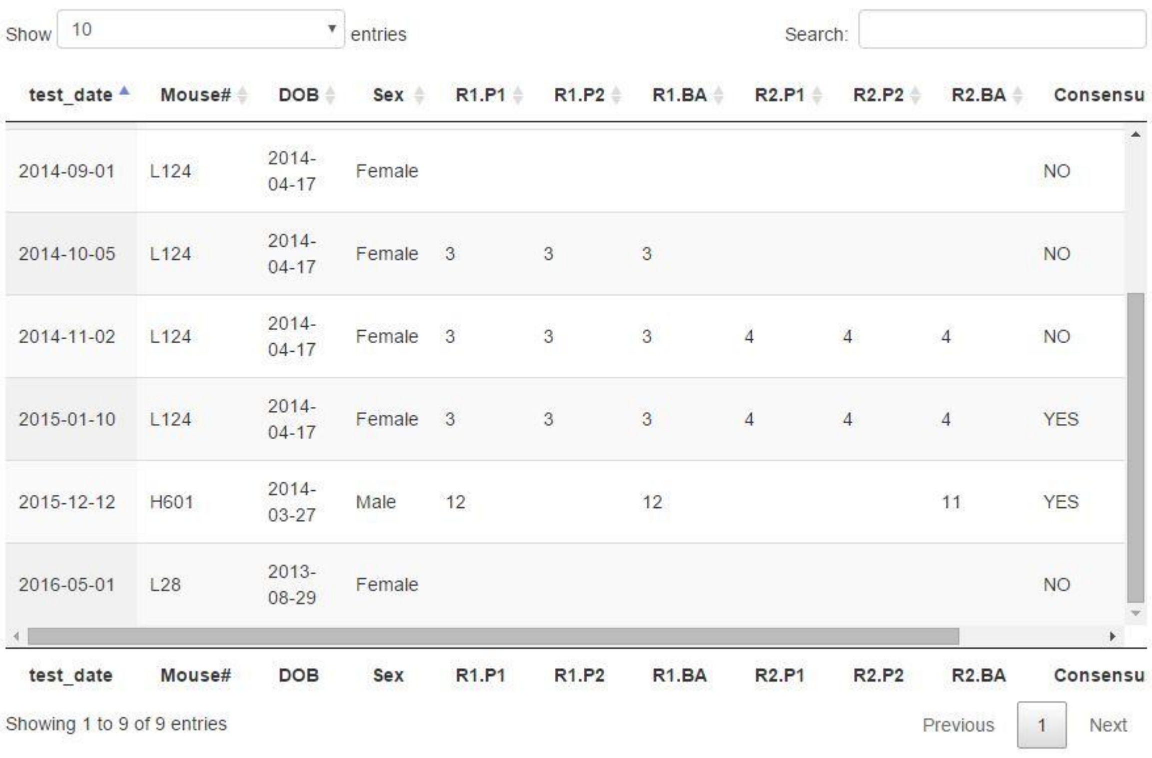
3.1 How many mice has been tested in different time period (3 months, 6 months, 9 months, 18 months is defined by client) and show its weight and data details.

3.2 How many mice has been tested in previous period but haven't go into the next part and show its weight.

Eg. Mouse Jimmy(mouse_id=001) has been tested in 3 months age but haven't been tested in 6 months.

Analysis




Show 10 entries Search:

test_date	Mouse#	DOB	Sex	R1.P1	R1.P2	R1.BA	R2.P1	R2.P2	R2.BA	Consensu
2014-09-01	L124	2014-04-17	Female							NO
2014-10-05	L124	2014-04-17	Female	3	3	3				NO
2014-11-02	L124	2014-04-17	Female	3	3	3	4	4	4	NO
2015-01-10	L124	2014-04-17	Female	3	3	3	4	4	4	YES
2015-12-12	H601	2014-03-27	Male	12		12			11	YES
2016-05-01	L28	2013-08-29	Female							NO

Showing 1 to 9 of 9 entries

Previous 1 Next

SQL:
//Mouse has been tested in 3 months not in 6 months
SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<105 and abs(DATEDIFF(test_date, DOB))>75
AND mouse_id NOT IN
(SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<195 and abs(DATEDIFF(test_date, DOB))>165);

//Mouse has been tested in 6 months not in 9 months

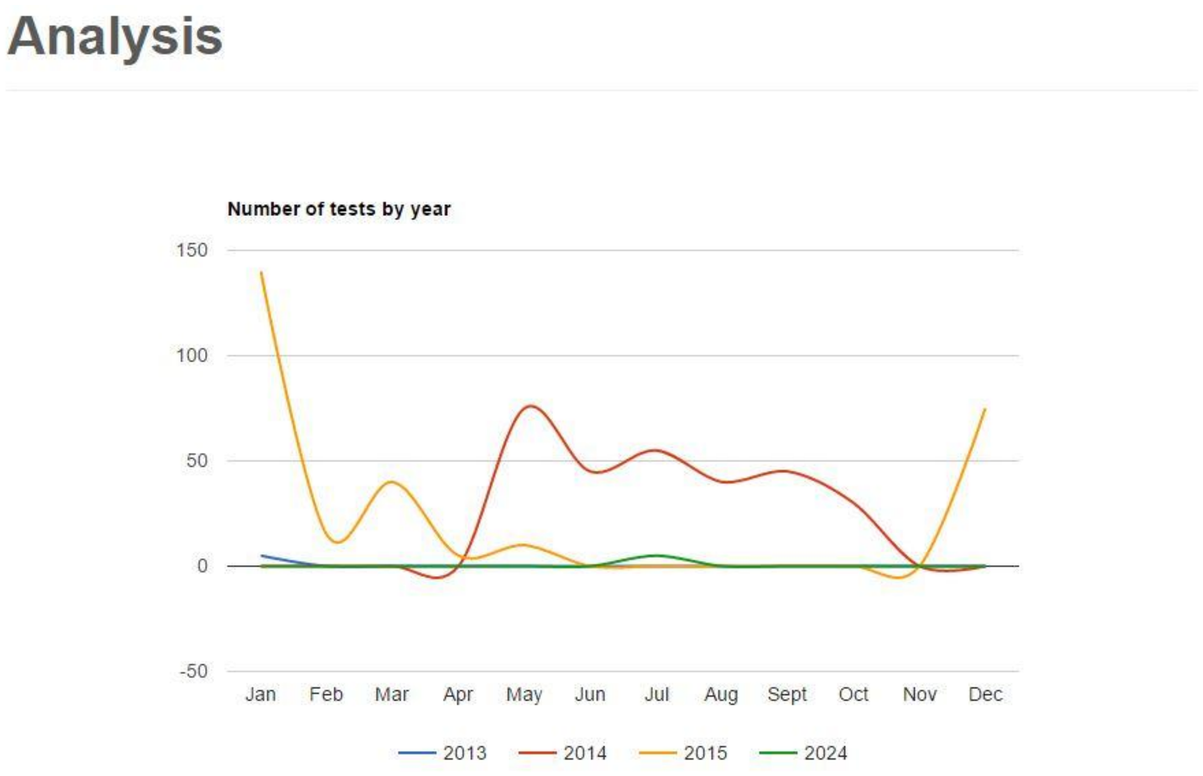
SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<195 and abs(DATEDIFF(test_date, DOB))>165
AND mouse_id NOT IN
(SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<285 and abs(DATEDIFF(test_date, DOB))>255);

//Mouse has been tested in 9 months not in 18 months
SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<285 and abs(DATEDIFF(test_date, DOB))>255
AND mouse_id NOT IN
(SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<555 and abs(DATEDIFF(test_date, DOB))>525);

//Mouse has been tested in 18 months
SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<555 and abs(DATEDIFF(test_date, DOB))>525;

//Mouse has been tested in other time period
SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<=75 or abs(DATEDIFF(test_date, DOB))>=555 or abs(DATEDIFF(test_date, DOB)) is NULL;

4. Frequency of tests in different year and month, which help the lab to generally understanding experiments history.

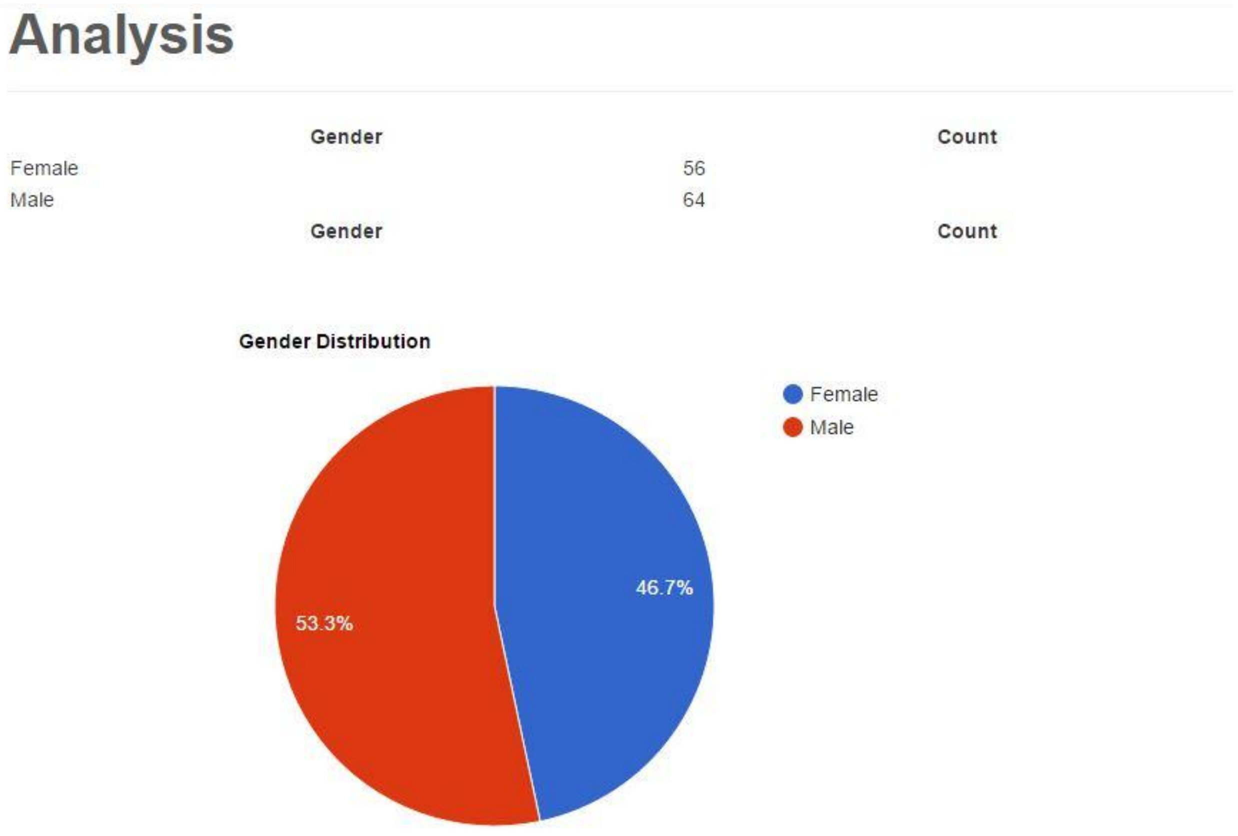


SQL:
//List number of tests for different month and year in increasing order
select count(*),MONTH(test_date),YEAR(test_date) from consensus group by

YEAR(test_date),MONTH(test_date) order by YEAR(test_date) ASC;

//List test years in increasing order
select distinct YEAR(test_date) from consensus order by YEAR(test_date);

5. Mice distribution by gender.



Analysis

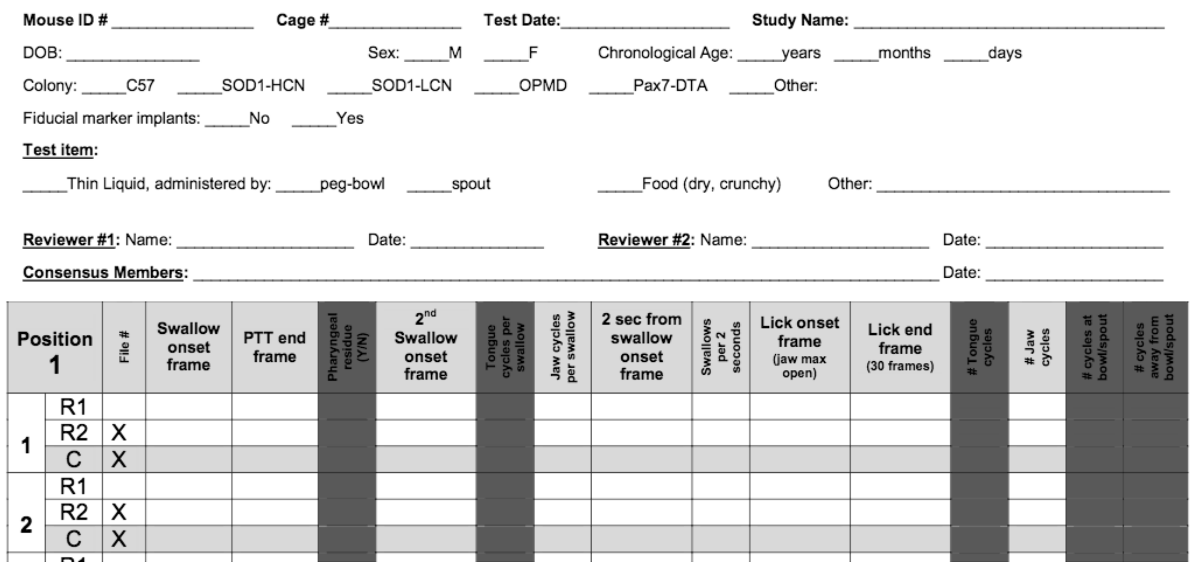
Gender	Count
Female	56
Male	64
Gender	Count

SQL:
//List mice number for different gender
SELECT sex, count(*) FROM subject GROUP BY sex;

5. Normalization

Because our database is used to store research data, we need to make sure the integrity of the data. In the class we have learned that redundancy can cause problems like insert/delete/update anomalies. When we designed the tables for our database, we tried to minimize data redundancy.

The following is part of one data collection sheet used by our client.



Mouse ID # ________________ **Cage #**_______________ **Test Date:**___________________ **Study Name:** ___________________________________

DOB: _______________ Sex: _____M _____F Chronological Age: _____years _____months _____days

Colony: _____C57 _____SOD1-HCN _____SOD1-LCN _____OPMD _____Pax7-DTA _____Other:

Fiducial marker implants: _____No _____Yes

Test item:

_____Thin Liquid, administered by: _____peg-bowl _____spout _____Food (dry, crunchy) Other: _________________________________

Reviewer #1: Name: ____________________ Date: _______________ **Reviewer #2**: Name: ____________________ Date: ____________________

Consensus Members: ____ Date: ____________________

Position 1		File #	**Swallow onset frame**	**PTT end frame**	Pharyngeal residue (Y/N)	**2nd Swallow onset frame**	Tongue cycles per swallow	Jaw cycles per swallow	**2 sec from swallow onset frame**	Swallows per 2 seconds	**Lick onset frame** (jaw max open)	**Lick end frame** (30 frames)	# Tongue cycles	# Jaw cycles	# cycles at bowl/spout	# cycles away from bowl/spout
1	R1															
	R2	X														
	C	X														
2	R1															
	R2	X														
	C	X														

We can see that for every experiment, besides the experimental data, they need to record Mouse ID, Cage #, DOB, Sex, Age, Colony type, Test Date, etc. The following functional dependencies hold between these attributes:

Mouse ID -> Cage #,
Mouse ID -> DOB,
Mouse ID -> Sex,
Mouse ID -> Colony type,
Mouse ID, Test Date -> Age

These functional dependencies indicate there are redundancy issues in their "paper-based database". The reason they use this recording method is that manually joining multiple tables are very time-consuming and error prone.

In our MySQL database, we have decomposed our client's data collection tables into several information tables that are in BCNF. Information tables include:

- student
- supervisor
- observer
- video
- publication
- subject
- colony
- human
- non_human
- cage
- pheno_type
- geno_type
- surgery
- treatment

We have three record tables (Pos1_record, Pos2_record, Bolus_record) to record raw experimental data for Position 1, Position 2, Bolus Area. They are in BCNF, too. These three tables share a same composite primary key: (m_id, trial_id, test_date, vid_id, reviewer_id). We chose the composite key due to following reasons:

- Each mouse may be tested more than once on a same day
- Each test has five trials for Position 1 and 2 and three trials for Bolus Area
- Each trial may have more than one videos recorded
- Each video will be reviewed by two reviewers and a consensus member

We also have one summary table "consensus", which stores the records that have been consented. The primary key for this table is a composite key (m_id, test_date, Trial). There are three functional dependencies in this table:

m_id, test_date -> age_month
age_month -> age_group
m_id, test_date -> age_group

As a result, this table is in 1NF. The reason we added "age_month" and "age_group" is that they are important attributes in a summary for our client.

6. Indexing selection

Due to the large amount of data in the database, we need to help client to hurry up their search or analysis query. The reason for the slow search speed can be reduced to 'order by', 'group by', '>,<,=', 'like', 'is null' clause. Thus, we includes some indexes in our database to speed up.
It contains: composite indexes and B-tree indexes.

1. create index test_date_index on consensus (test_date);
Reason:
select distinct YEAR(test_date) from consensus order by YEAR(test_date);

select count(*),MONTH(test_date),YEAR(test_date) from consensus group by YEAR(test_date),MONTH(test_date) order by YEAR(test_date) ASC;

2. create index tdate_dob_index on review_record(test_date, DOB);
Reason:
SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<105 and abs(DATEDIFF(test_date, DOB))>75
AND mouse_id NOT IN
(SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<195 and abs(DATEDIFF(test_date, DOB))>165);

SELECT mouse_id from review_record where abs(DATEDIFF(test_date, DOB))<=75 or abs(DATEDIFF(test_date, DOB))>=555 or abs(DATEDIFF(test_date, DOB)) is NULL;

3. create index sex_index on subject (sex);
Reason:
SELECT sex, count(*) FROM subject GROUP BY sex;

4. create index id_age_index on consensus (m_id,age_month) using btree;
Reason:
select distinct(age) from (select age,name,avg(rate) from (select m_id,avg(Jaw_rate) as rate,test_date,round(age_month,0) as age,name from consensus join (select sub_id,name from geno_type join (select * from subject natural join non_human_view natural join has_geno) as test on (geno_type.geno_id=test.geno_id))as geno on (geno.sub_id=consensus.m_id) group by m_id, age_month) as s group by s.age,s.name) as s2;

5. create index geno_id_index on geno_type (geno_id) using btree;
Reason:
select sex,geno_type.geno_id,geno_type.name,count(*) from geno_type left join (select * from subject natural join non_human_view natural join has_geno) as test on (test.geno_id=geno_type.geno_id) group by geno_type.geno_id,sex;

6. create index id_index on mouse (sub_id);
Reason:
SELECT sub_id FROM mouse WHERE sub_id LIKE 'H%';

7. create index rate_id_index on consensus (Jaw_rate,m_id);
Reason:
SELECT DISTINCT(m_id), count(*) FROM consensus WHERE Jaw_rate IS NULL group by (m_id);

7. Optimization and Tuning

In the class, we have learned that the choice of conceptual schema should be guided by the workload, in addition to redundancy issues. There are many ways to tune the conceptual schema. We may need to choose a normal form other than BCNF or further decompose a BCNF schema. We may denormalize or add fields to a relation. We may consider horizontal decompositions.

When we designed our system, we considered tuning.

The consensus table contains three functional dependencies:

m_id, test_date -> age_month
age_month -> age_group
m_id, test_date -> age_group

If we know the ID of one mouse, we can get its DOB from the “subject” using a join and a simple arithmetic calculation (Test date - DOB) can give us the age in months and the corresponding age group. However, we know that arithmetic calculation is not efficient in

MySQL and it cannot benefit from the indices on the attributes being used. What's more, age in month and age group are important attributes for our client and should be included in the consensus table.

This table could have been in BCNF without age_month and age_group. But considering the workload and our client's preference, we decided to keep age_month and age_group in the consensus table. As a result, the table becomes 1NF.

We have also used horizontal decomposition for non-human subjects. Besides mouse, our client has other types of animal models, such as dog, pig, and horse. Most of their experiments are done on mice and only a few are on other models. We used to have a "non_human" table, which stores all animal models and they are differentiated by an attribute "category". Querying for models other than mouse will need to go through the whole table, which is not efficient.

To address this issue, we have horizontally decomposed the "non_human" table into "mouse", "dog", "pig", and "horse". A view "non_human_view" has been created to mask this change. The SQL command is:

```
CREATE VIEW non_human_view AS
SELECT * FROM mouse
UNION
SELECT * FROM dog
UNION
SELECT * FROM pig
UNION
SELECT * FROM horse;
```

8. Security setting

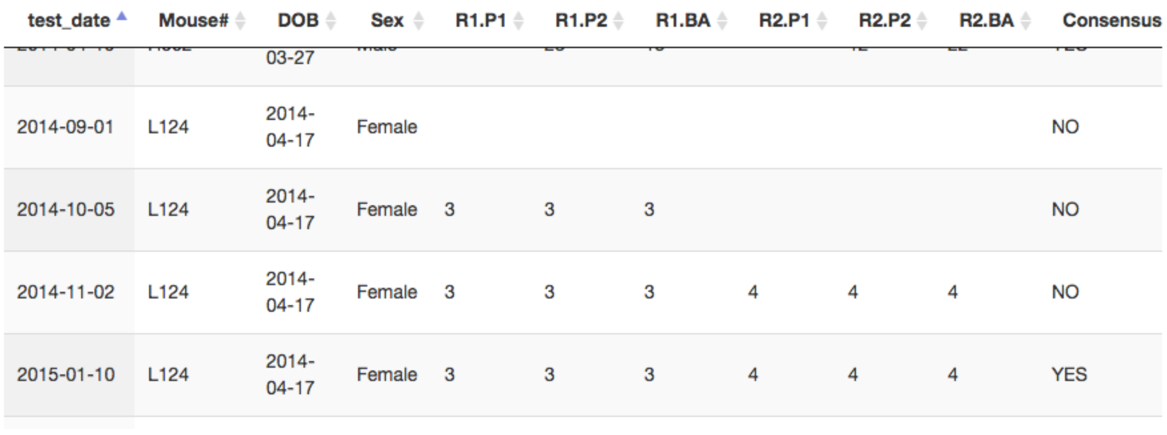
In the class, we have learned that we need to ensure secrecy, integrity, availability, which means users should not be able to see or modify things they are not supposed to and they should be able to see and modify things they are allowed to. There are two main mechanism at the DBMS level, including discretionary access control and mandatory access control.

In our database design, we adopted the discretionary access control mechanism, which is based on the concept of access rights and privileges for objects and mechanisms for giving users privileges.

There are two types of accounts in our database, including supervisor account and student account. Dr. Lever and her lab assistant Kate have supervisor accounts. The students in Dr. Lever's group have student accounts no matter they are graduate or undergraduate students. Dr. Lever and Kate have all privileges on all tables in the database. Beyond these, Dr. Lever can create new user accounts and can grant privileges. We chose this design because we want to make sure Dr. Lever can keep control of everything. But we can always make changes if Dr. Lever wants to delegate these privileges to Kate.

Students are not allowed to create accounts by themselves. Because our database is used to manage experimental data, it’s very important that students are not allowed to delete or update data to ensure data safety and research ethics. Students are not allowed to view “observer”, “student”, and “supervisor” tables since they contain information of other accounts. These three tables can only be accessed by supervisor accounts. What’s more, students should be blinded when they insert raw experimental data (the second reviewer should not be able to see data recorded by the first reviewer to make sure there is no bias). Students are allowed to insert the raw experimental data into “Pos1_record”, “Pos2_record”, and “Bolus_record” table. Students are allowed to view the rest tables.

We have also created a view (“review_view”) to summarize the research progress. It can show if the videos have been reviewed or consented. It only shows the IDs of the reviewers and consensus status and hides the underlying information of each table. This can make sure that students can see the whole progress but cannot see the raw data recorded by other students.



test_date	Mouse#	DOB	Sex	R1.P1	R1.P2	R1.BA	R2.P1	R2.P2	R2.BA	Consensus
[illegible]	[illegible]	03-27	[illegible]		[illegible]	[illegible]		[illegible]	[illegible]	[illegible]
2014-09-01	L124	2014-04-17	Female							NO
2014-10-05	L124	2014-04-17	Female	3	3	3				NO
2014-11-02	L124	2014-04-17	Female	3	3	3	4	4	4	NO
2015-01-10	L124	2014-04-17	Female	3	3	3	4	4	4	YES

The above picture shows Mouse L124 was tested the first time on 2014-09-01 and has not been reviewed by any students. On 2014-10-05, L124 was tested again and only one student (with ID = 3) has reviewed the video and recorded the raw data for position 1, position 2 and bolus area. On 2014-11-02, L124 was tested and this time two students (with ID = 3 and 4) have reviewed the video and recorded the raw data. But the raw data has not been consented yet (Consensus = NO). On 2015-01-10, L124 was tested again and this time the video has been reviewed by two students (with ID = 3 and 4) and has been consented, too (Consensus = YES).

The SQL command to create this view is:

```
CREATE VIEW review_view AS
SELECT DISTINCT V.test_date, S.sub_id, S.dob, S.sex,
        IF(S.sub_id IN (SELECT P1.m_id FROM Pos1_record P1 WHERE P1.test_date = V.test_date AND P1.m_id = S.sub_id AND P1.status = 'r1'), (SELECT P1.reviewer_id FROM Pos1_record P1 WHERE P1.test_date = V.test_date AND P1.m_id = S.sub_id AND P1.status = 'r1'), ' ') AS 1stP1,
        IF(S.sub_id IN (SELECT P1.m_id FROM Pos1_record P1 WHERE P1.test_date = V.test_date AND P1.m_id = S.sub_id AND P1.status = 'r2'), (SELECT P1.reviewer_id FROM
Pos1_record P1 WHERE P1.test_date = V.test_date AND P1.m_id = S.sub_id AND P1.status = 'r2'), ' ') AS 2ndP1,
        IF(S.sub_id IN (SELECT P2.m_id FROM Pos2_record P2 WHERE P2.test_date = V.test_date AND P2.m_id = S.sub_id AND P2.status = 'r1'), (SELECT P2.reviewer_id FROM Pos2_record P2 WHERE P2.test_date = V.test_date AND P2.m_id = S.sub_id AND P2.status = 'r1'), ' ') AS 1stP2,
        IF(S.sub_id IN (SELECT P2.m_id FROM Pos2_record P2 WHERE P2.test_date = V.test_date AND P2.m_id = S.sub_id AND P2.status = 'r2'), (SELECT P2.reviewer_id FROM Pos2_record P2 WHERE P2.test_date = V.test_date AND P2.m_id = S.sub_id AND P2.status = 'r2'), ' ') AS 2ndP2,
        IF(S.sub_id IN (SELECT B.m_id FROM Bolus_record B WHERE B.test_date = V.test_date AND B.m_id = S.sub_id AND B.status = 'r1'), (SELECT B.reviewer_id FROM Bolus_record B WHERE B.test_date = V.test_date AND B.m_id = S.sub_id AND B.status = 'r1'), ' ') AS 1stBA,
        IF(S.sub_id IN (SELECT B.m_id FROM Bolus_record B WHERE B.test_date = V.test_date AND B.m_id = S.sub_id AND B.status = 'r2'), (SELECT B.reviewer_id FROM Bolus_record B WHERE B.test_date = V.test_date AND B.m_id = S.sub_id AND B.status = 'r2'), ' ') AS 2ndBA,
        IF(S.sub_id IN (SELECT P1.m_id FROM Pos1_record P1, Pos2_record P2 WHERE P2.m_id = P1.m_id AND P1.m_id = S.sub_id AND P1.test_date = V.test_date AND P2.test_date = V.test_date AND P1.status = 'c' AND P2.status = 'c'), 'YES', 'NO') AS consensus
FROM video V, Pos1_record P1, Pos2_record P2, Bolus_record B, subject S, has_vid H
WHERE V.vid_id = H.vid_id AND H.sub_id = S.sub_id;
```

9. Other topics

9.1 Triggers

We have created several triggers to archive information. For example, if a student graduates and leaves Dr. Lever's group. His/her account will be deleted and saved to an archived table.

The codes to create the triggers are:

<u>To archive deleted accounts:</u>

```
DELIMITER $$
CREATE TRIGGER archive_deleted_accounts
BEFORE DELETE ON observer
FOR EACH ROW
BEGIN
        DECLARE detail VARCHAR(20);
        IF EXISTS (SELECT * FROM student WHERE obs_id = OLD.obs_id AND type = “graduate”) THEN SET detail = ‘graduate student’;
        ELSEIF EXISTS (SELECT * FROM student WHERE obs_id = OLD.obs_id AND type = “undergraduate”) THEN SET detail = ‘undergraduate student’;
        ELSEIF EXISTS (SELECT * FROM supervisor WHERE obs_id = OLD.obs_id) THEN
        SET detail = ‘supervisor’;
        END IF;
        INSERT INTO archived_observer;
                SET           obs_id=OLD.obs_id,           first_name=OLD.first_name,
                last_name=OLD.last_name, password=OLD.password, description = detail;
END $$
DELIMITER ;
```

<u>To archive deleted genotypes:</u>

```
DELIMITER $$
CREATE TRIGGER archive_deleted_genotypes
BEFORE DELETE ON geno_type
FOR EACH ROW
BEGIN
        INSERT INTO archived_geno_type;
        SET geno_id=OLD.geno_id, name=OLD.name, description=OLD.description;
END $$
DELIMITER ;
```

<u>To archive deleted phenotypes:</u>

```
DELIMITER $$
CREATE TRIGGER archive_deleted_phenotypes
BEFORE DELETE ON pheno_type
FOR EACH ROW
BEGIN
        INSERT INTO archived_pheno_type;
        SET pheno_id=OLD.pheno_id, name=OLD.name, description=OLD.description;
END $$
DELIMITER ;
```

<u>To archive deleted colonies:</u>

```
DELIMITER $$
CREATE TRIGGER archive_deleted_colonies
BEFORE DELETE ON colony
FOR EACH ROW
BEGIN
        INSERT INTO archived_colony;
        SET col_id=OLD.col_id, disease=OLD.disease, CNV=OLD.CNV;
END $$
DELIMITER ;
```

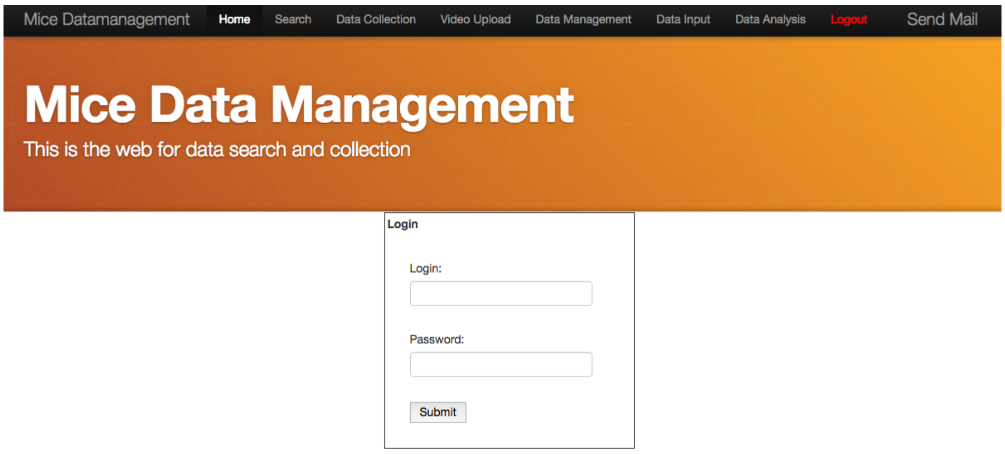
9.2 Data cleaning

We have done some data cleaning and conversion. The sample data we got from our client is in .sav format, which is a proprietary format of SPSS. We first exported the .sav file to a csv file. Then some data cleaning was done to make the csv file compatible with MySQL.

10. User Manual

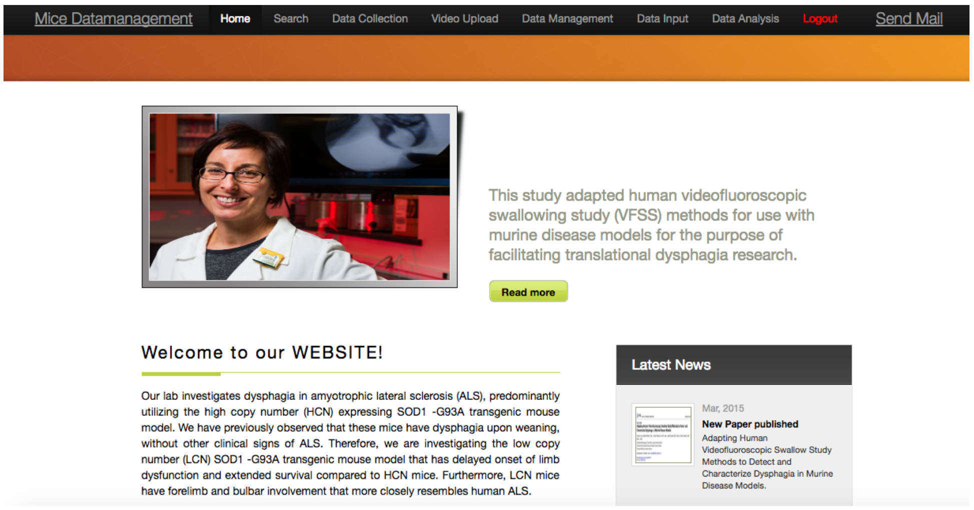
10.1. Login

There are two types of users for the website. One is student, the other is supervisor. Correct username and password are required to login.



10.2. Home Page

After login, user can go to our home page. The homepage contains the basic introduction to our client’s experiment, published papers and some latest news.



This study adapted human videofluoroscopic swallowing study (VFSS) methods for use with murine disease models for the purpose of facilitating translational dysphagia research.

Welcome to our WEBSITE!

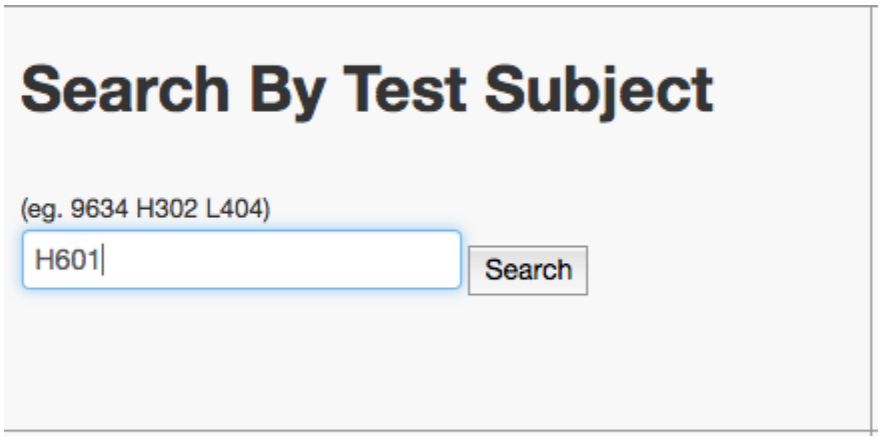
Our lab investigates dysphagia in amyotrophic lateral sclerosis (ALS), predominantly utilizing the high copy number (HCN) expressing SOD1 -G93A transgenic mouse model. We have previously observed that these mice have dysphagia upon weaning, without other clinical signs of ALS. Therefore, we are investigating the low copy number (LCN) SOD1 -G93A transgenic mouse model that has delayed onset of limb dysfunction and extended survival compared to HCN mice. Furthermore, LCN mice have forelimb and bulbar involvement that more closely resembles human ALS.

Mar, 2015
New Paper published
Adapting Human Videofluoroscopic Swallow Study Methods to Detect and Characterize Dysphagia in Murine Disease Models.

10.3. Search

This page is convenient for research analysis. Based on our client’s requirement, we create 4 kinds of search.

Search by test subject:

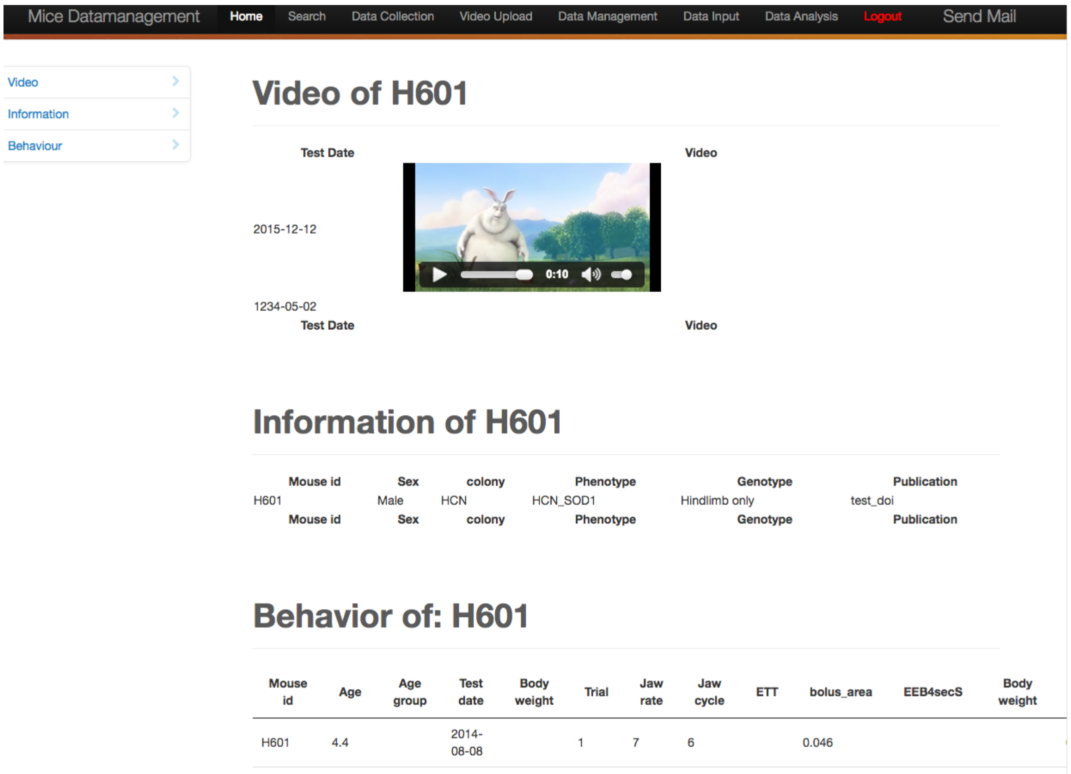


Search By Test Subject

(eg. 9634 H302 L404)

H601

By input the test subject, it will pull out all the information about this subject.



Video of H601

Test Date	Video
2015-12-12	
1234-05-02	
Test Date	Video

Information of H601

Mouse id	Sex	colony	Phenotype	Genotype	Publication
H601	Male	HCN	HCN_SOD1	Hindlimb only	test_doi
Mouse id	Sex	colony	Phenotype	Genotype	Publication

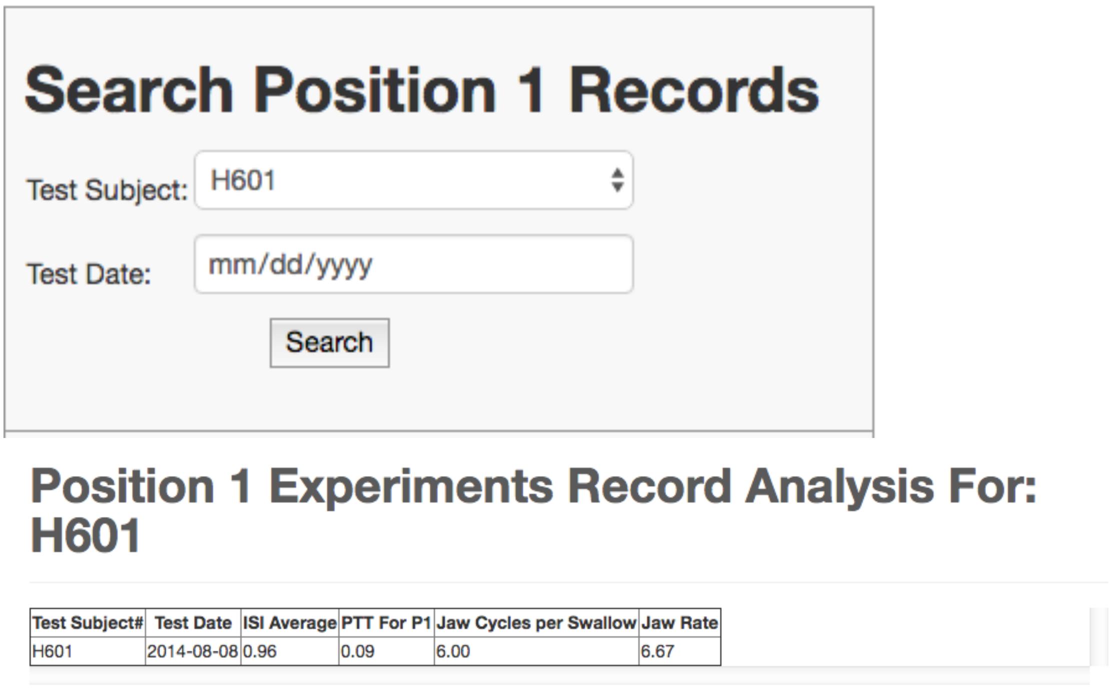
Behavior of: H601

Mouse id	Age	Age group	Test date	Body weight	Trial	Jaw rate	Jaw cycle	ETT	bolus_area	EEB4secS	Body weight
H601	4.4		2014-08-08		1	7	6		0.046		

Search analytical data for position 1’s record:

We provide some basic data analysis here for all the position 1’s record. Such as average PTT time, ISI time for each test subject on one test date.

You can search one test subject; this will pull out all the result from this test subject on every test day. Also, you can search one specific day, this will show all experiments test subject’s data on this day.



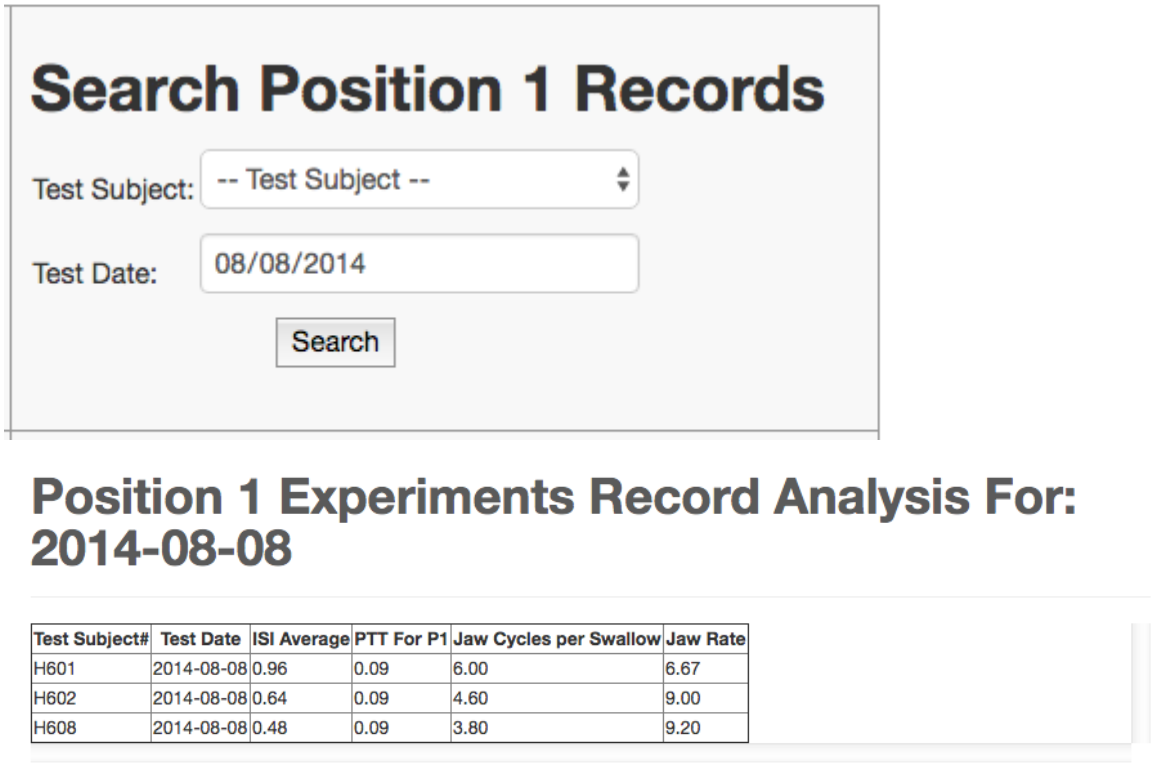
Search Position 1 Records

Test Subject: H601

Test Date: mm/dd/yyyy

Position 1 Experiments Record Analysis For: H601

Test Subject#	Test Date	ISI Average	PTT For P1	Jaw Cycles per Swallow	Jaw Rate
H601	2014-08-08	0.96	0.09	6.00	6.67



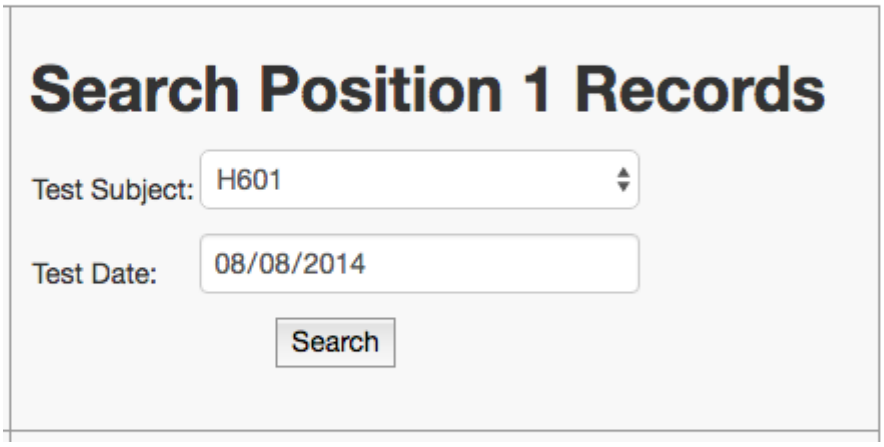
Search Position 1 Records

Test Subject: -- Test Subject --

Test Date: 08/08/2014

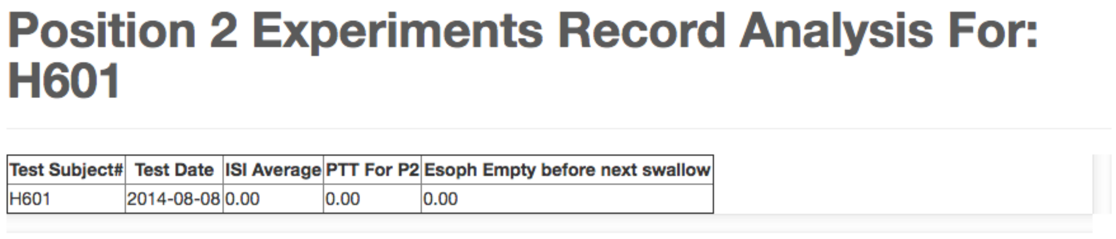
Position 1 Experiments Record Analysis For: 2014-08-08

Test Subject#	Test Date	ISI Average	PTT For P1	Jaw Cycles per Swallow	Jaw Rate
H601	2014-08-08	0.96	0.09	6.00	6.67
H602	2014-08-08	0.64	0.09	4.60	9.00
H608	2014-08-08	0.48	0.09	3.80	9.20

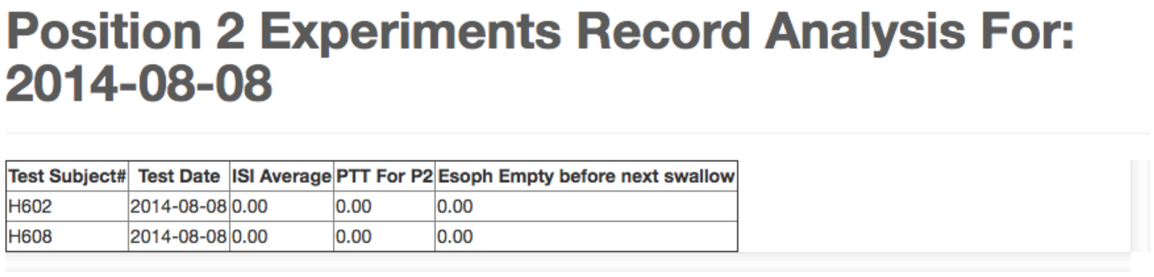

Search Position 1 Records

Test Subject:	H601
Test Date:	08/08/2014

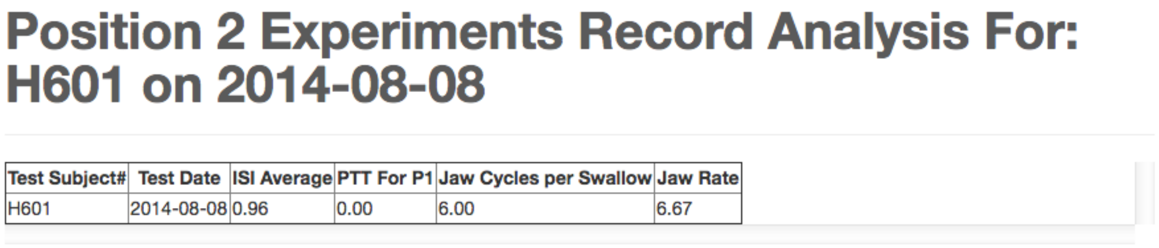
Search analytical data for position 2's record:
This search is similar with the search result for position 1. But different data analysis for position2.


Position 2 Experiments Record Analysis For: H601

Test Subject#	Test Date	ISI Average	PTT For P2	Esoph Empty before next swallow
H601	2014-08-08	0.00	0.00	0.00

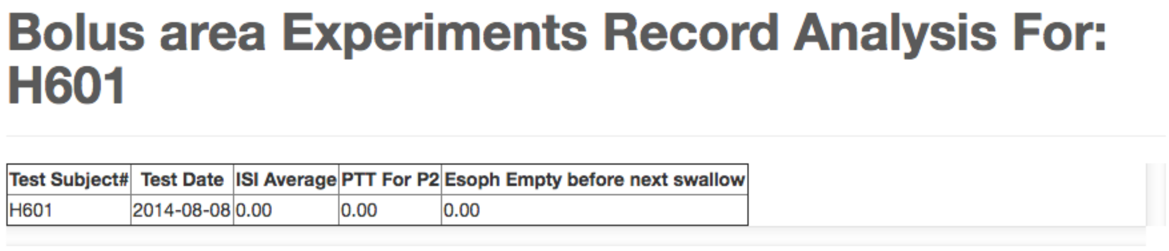

Position 2 Experiments Record Analysis For: 2014-08-08

Test Subject#	Test Date	ISI Average	PTT For P2	Esoph Empty before next swallow
H602	2014-08-08	0.00	0.00	0.00
H608	2014-08-08	0.00	0.00	0.00


Position 2 Experiments Record Analysis For: H601 on 2014-08-08

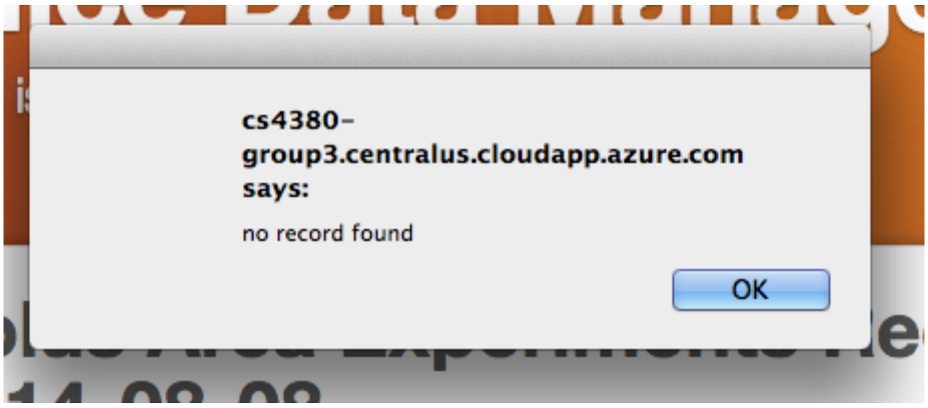
Test Subject#	Test Date	ISI Average	PTT For P1	Jaw Cycles per Swallow	Jaw Rate
H601	2014-08-08	0.96	0.00	6.00	6.67

Search analytical data for bolus area's record:
This search is similar with the search result for position 1 as well. But different data calculation is applied for bolus area.


Bolus area Experiments Record Analysis For: H601

Test Subject#	Test Date	ISI Average	PTT For P2	Esoph Empty before next swallow
H601	2014-08-08	0.00	0.00	0.00

If no record found, it will show an alert and go back to search page.



cs4380-group3.centralus.cloudapp.azure.com says:

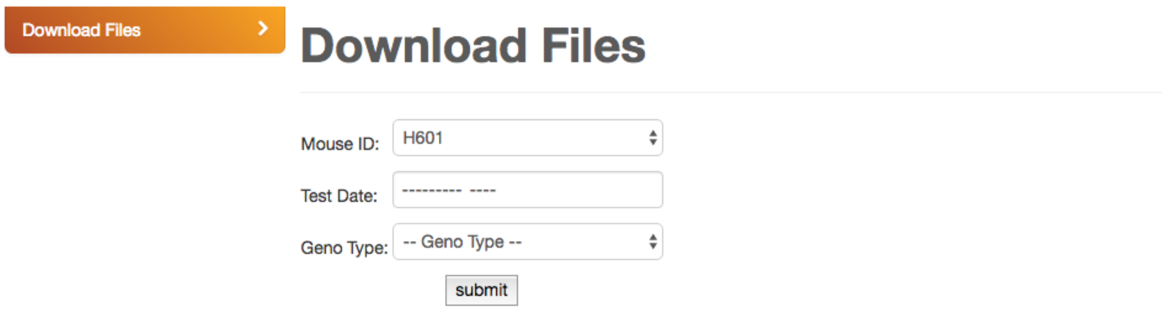
no record found

Download csv files:

We provide downloadable csv files for all data. If our client want to do further research based on the experiment data, they can download csv files to mining more results.

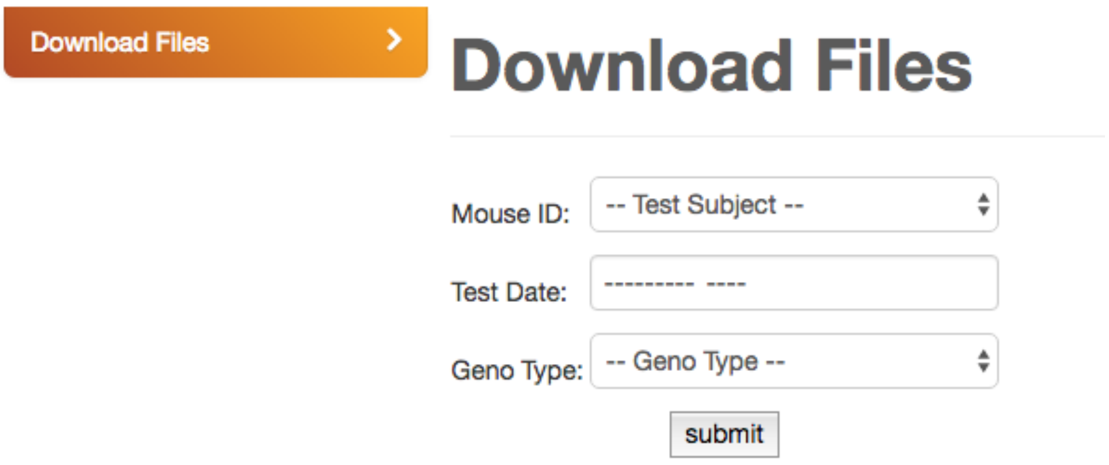
There are 3 types of queries that can download the files:

You can down all experiment result for one specific test subject, one specific test month, one genotype related test subject.



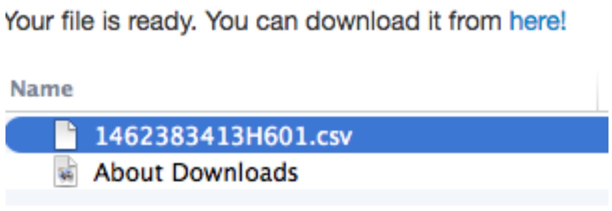
Download Files

Mouse ID:	H601
Test Date:	--------- ----
Geno Type:	-- Geno Type --



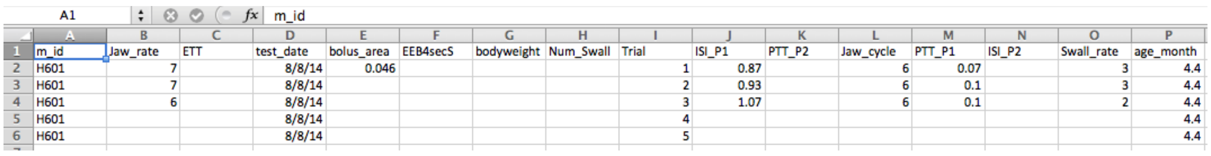
Download Files

Mouse ID:	-- Test Subject --
Test Date:	--------- ----
Geno Type:	-- Geno Type --



Your file is ready. You can download it from here!

Name
1462383413H601.csv
About Downloads

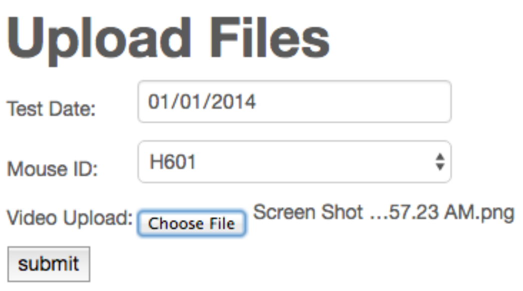


A1 m_id

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	m_id	Jaw_rate	ETT	test_date	bolus_area	EEB4secS	bodyweight	Num_Swall	Trial	ISI_P1	PTT_P2	Jaw_cycle	PTT_P1	ISI_P2	Swall_rate	age_month
2	H601	7		8/8/14	0.046				1	0.87		6	0.07		3	4.4
3	H601	7		8/8/14					2	0.93		6	0.1		3	4.4
4	H601	6		8/8/14					3	1.07		6	0.1		2	4.4
5	H601			8/8/14					4							4.4
6	H601			8/8/14					5							4.4

10.4. Video Upload

After one experiment has down. Our client can upload video file to the server for reviewer watch the video. The video is specified by one test date and one test subject

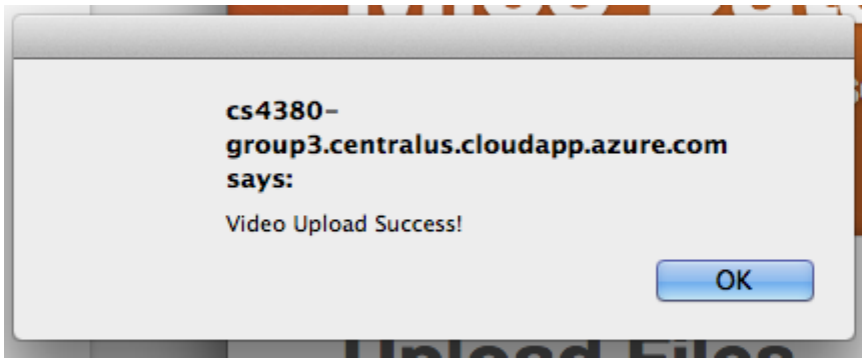


Upload Files

Test Date: 01/01/2014

Mouse ID: H601

Video Upload: Choose File Screen Shot ...57.23 AM.png



cs4380-group3.centralus.cloudapp.azure.com says:

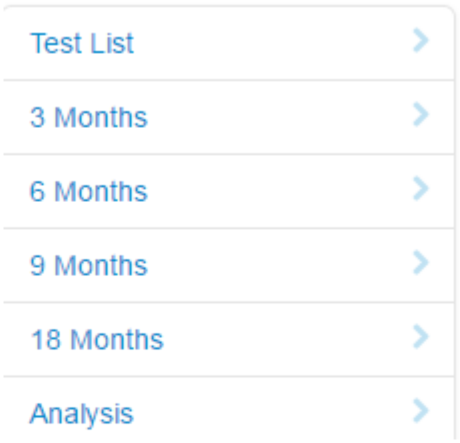
Video Upload Success!

10.5. Data Management

10.6. Data Collection

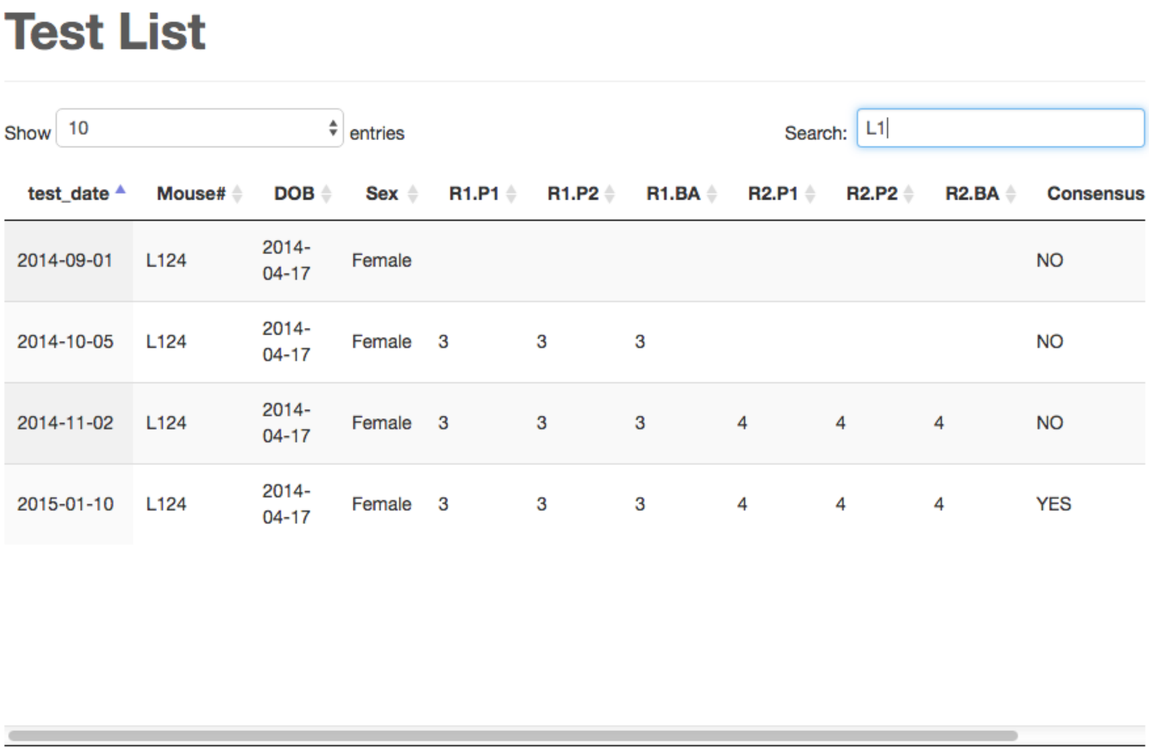
10.6.1. Experiments Check List

Check list for all the experiments that have been done. Since there is hundreds of experiments have done, we separate the checklist into different test subject's age.



After one experiment has done, it needs to be reviewed by 2 students and students will record the corresponding data and timestamp for each trail of the experiment. The experiments will also need to be checked by the supervisor faculty.

For the checklist, each experiment has 3 optional statuses: reviewed once, reviewed twice, consensus. The checklist will record the name of reviewers' name. If the experiment has not been consensus yet, it needs to be checked by supervisor to consensus the two reviewers' record.
Reviewer can also search mouse's record by their id on the top. Since there is too many test subjects, we provide fuzzy search by mouse id to find one specific test subject.



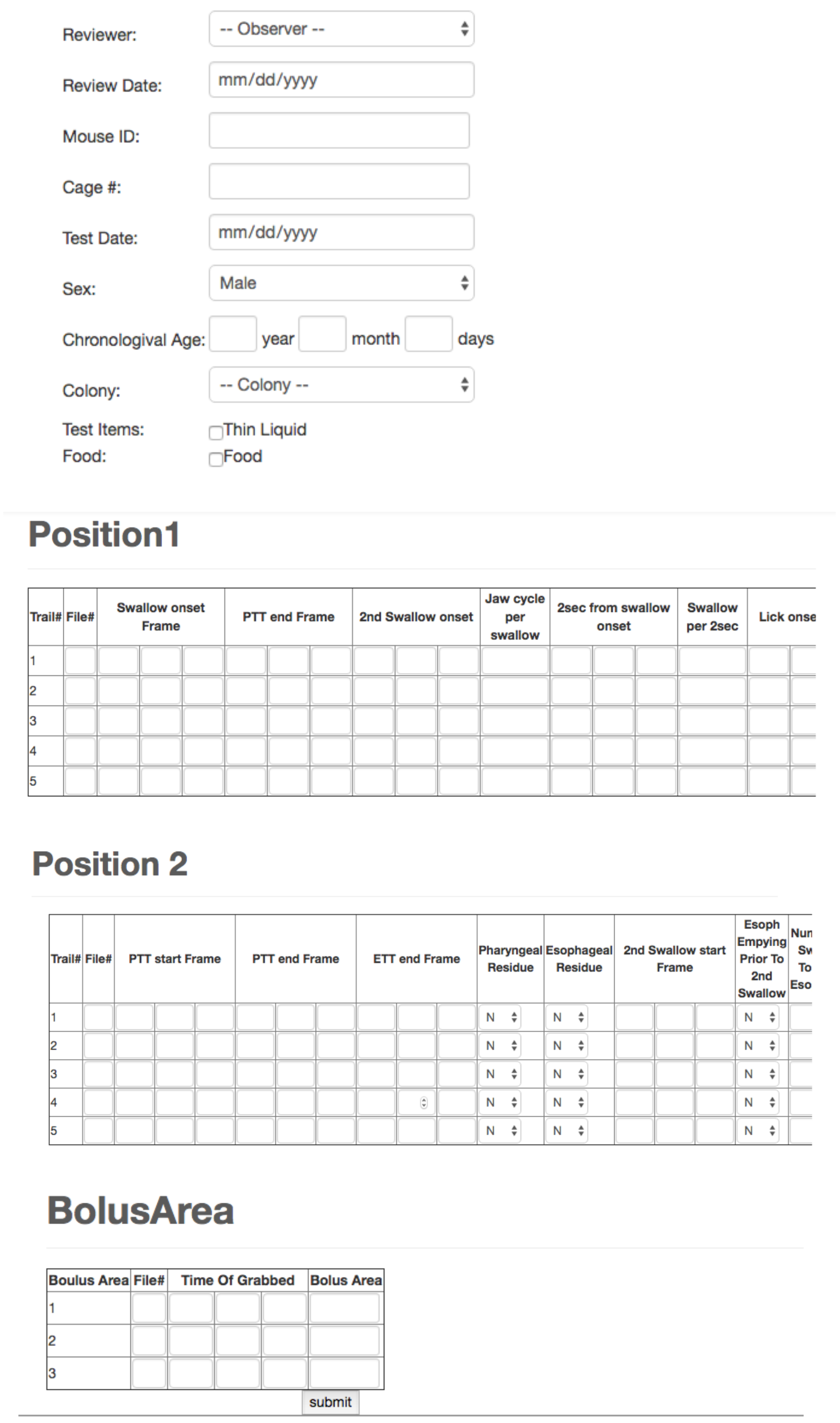
Test List

Show 10 entries Search: L1

test_date	Mouse#	DOB	Sex	R1.P1	R1.P2	R1.BA	R2.P1	R2.P2	R2.BA	Consensus
2014-09-01	L124	2014-04-17	Female							NO
2014-10-05	L124	2014-04-17	Female	3	3	3				NO
2014-11-02	L124	2014-04-17	Female	3	3	3	4	4	4	NO
2015-01-10	L124	2014-04-17	Female	3	3	3	4	4	4	YES

10.6.2. Experiments Record

For reviewer 1: there is a link to one experiment's video. Reviewer 1 will watch the video and take down the corresponding timestamp and experiment data for each trail of the experiment. Reviewer 1 also needs to record the



Reviewer:	-- Observer --
Review Date:	mm/dd/yyyy
Mouse ID:	
Cage #:	
Test Date:	mm/dd/yyyy
Sex:	Male
Chronologival Age:	year month days
Colony:	-- Colony --
Test Items:	☐ Thin Liquid
Food:	☐ Food

Position1

Trail#	File#	Swallow onset Frame	PTT end Frame	2nd Swallow onset	Jaw cycle per swallow	2sec from swallow onset	Swallow per 2sec	Lick onse
1								
2								
3								
4								
5								

Position 2

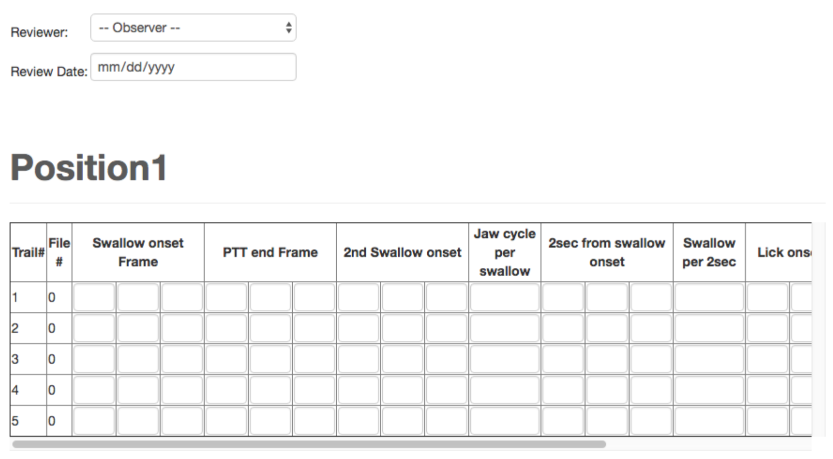
Trail#	File#	PTT start Frame	PTT end Frame	ETT end Frame	Pharyngeal Residue	Esophageal Residue	2nd Swallow start Frame	Esoph Empying Prior To 2nd Swallow	Nun Sw To Eso
1					N	N		N	
2					N	N		N	
3					N	N		N	
4					N	N		N	
5					N	N		N	

BolusArea

Boulus Area	File#	Time Of Grabbed	Bolus Area
1			
2			
3			

submit

For reviewer 2: there also a link to the same video as reviewer 1. Reviewer 2 will get some information like the file number, this is automatically shown on the website. Then reviewer 2 will record the experiment data again, without looking reviewer 1's result.

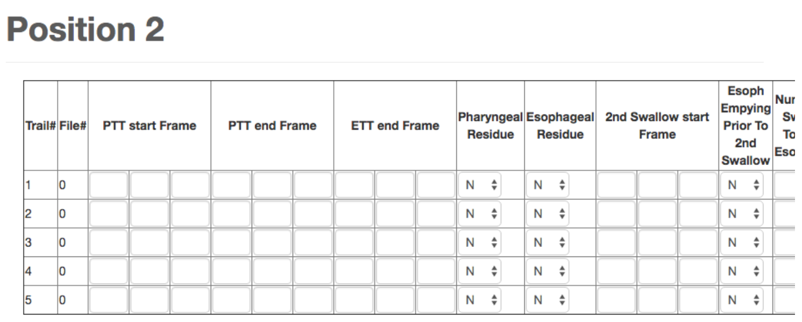


Reviewer: -- Observer --

Review Date: mm/dd/yyyy

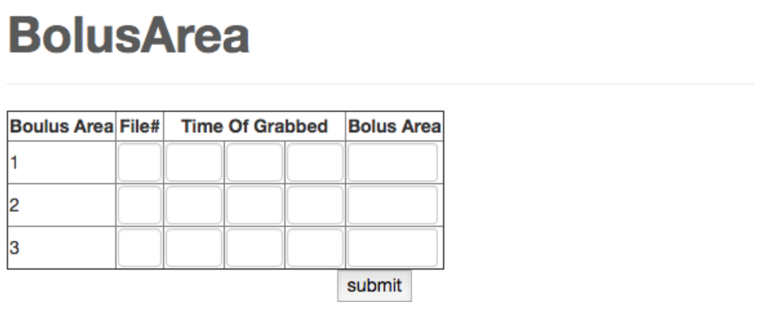
Position1

Trail#	File #	Swallow onset Frame	PTT end Frame	2nd Swallow onset	Jaw cycle per swallow	2sec from swallow onset	Swallow per 2sec	Lick ons
1	0							
2	0							
3	0							
4	0							
5	0							



Position 2

Trail#	File#	PTT start Frame	PTT end Frame	ETT end Frame	Pharyngeal Residue	Esophageal Residue	2nd Swallow start Frame	Esoph Empying Prior To 2nd Swallow	Nur Sw To Eso
1	0				N	N		N	
2	0				N	N		N	
3	0				N	N		N	
4	0				N	N		N	
5	0				N	N		N	



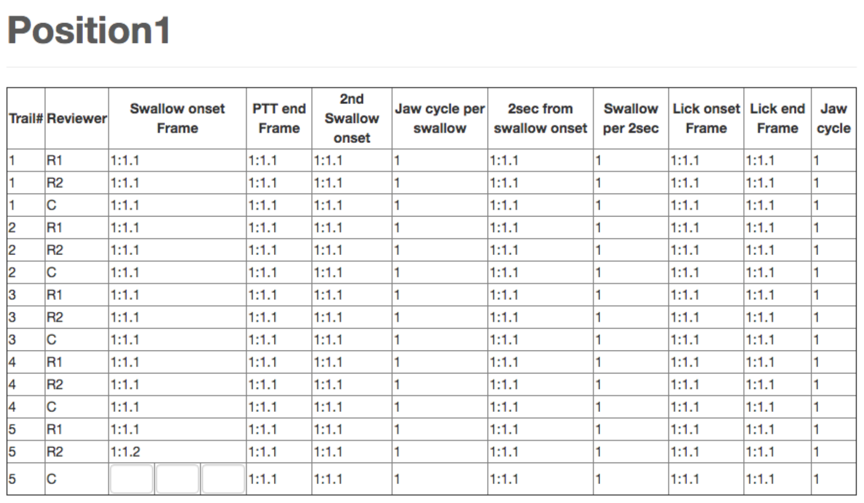
BolusArea

Boulus Area	File#	Time Of Grabbed	Bolus Area
1			
2			
3			

submit

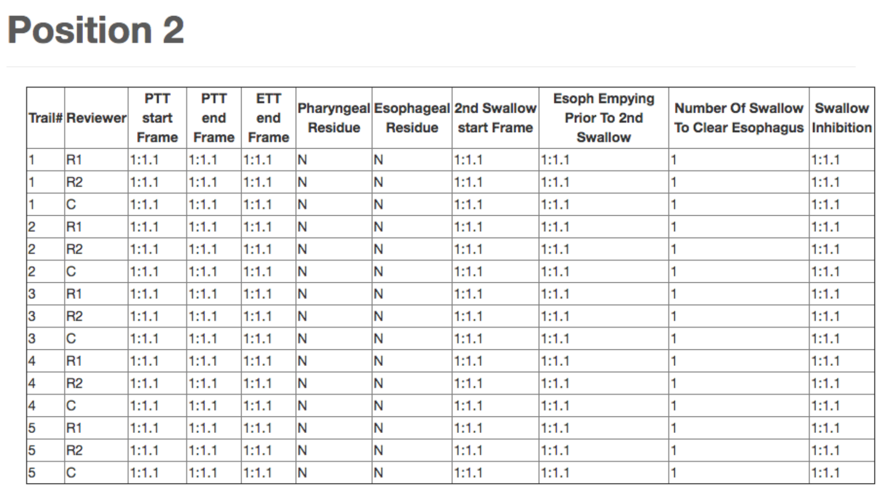
10.6.3. Experiments Consensus

For consensus member, the whole record from both reviewer 1 and reviewer 2, if they are not same, consensus member will need to watch the video again and make a third review of the video, and then take down the consensus data.



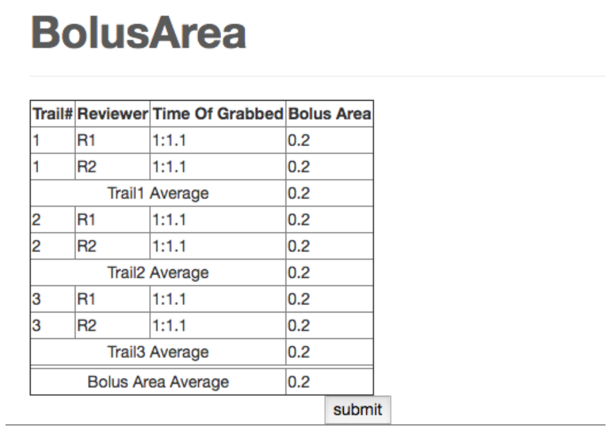
Position1

Trail#	Reviewer	Swallow onset Frame	PTT end Frame	2nd Swallow onset	Jaw cycle per swallow	2sec from swallow onset	Swallow per 2sec	Lick onset Frame	Lick end Frame	Jaw cycle
1	R1	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
1	R2	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
1	C	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
2	R1	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
2	R2	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
2	C	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
3	R1	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
3	R2	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
3	C	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
4	R1	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
4	R2	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
4	C	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
5	R1	1:1.1	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
5	R2	1:1.2	1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1
5	C		1:1.1	1:1.1	1	1:1.1	1	1:1.1	1:1.1	1



Position 2

Trail#	Reviewer	PTT start Frame	PTT end Frame	ETT end Frame	Pharyngeal Residue	Esophageal Residue	2nd Swallow start Frame	Esoph Empying Prior To 2nd Swallow	Number Of Swallow To Clear Esophagus	Swallow Inhibition
1	R1	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
1	R2	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
1	C	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
2	R1	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
2	R2	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
2	C	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
3	R1	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
3	R2	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
3	C	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
4	R1	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
4	R2	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
4	C	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
5	R1	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
5	R2	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1
5	C	1:1.1	1:1.1	1:1.1	N	N	1:1.1	1:1.1	1	1:1.1

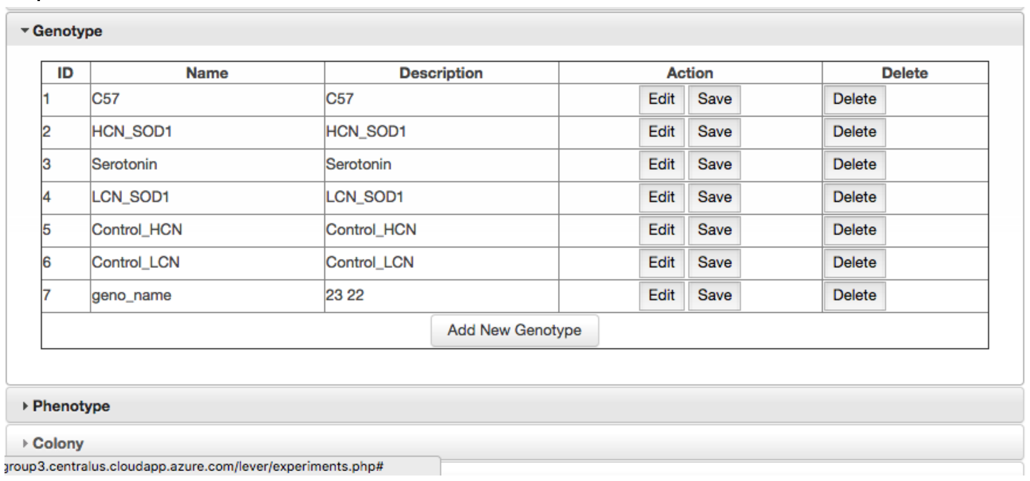


BolusArea

Trail#	Reviewer	Time Of Grabbed	Bolus Area
1	R1	1:1.1	0.2
1	R2	1:1.1	0.2
Trail1 Average			0.2
2	R1	1:1.1	0.2
2	R2	1:1.1	0.2
Trail2 Average			0.2
3	R1	1:1.1	0.2
3	R2	1:1.1	0.2
Trail3 Average			0.2
Bolus Area Average			0.2

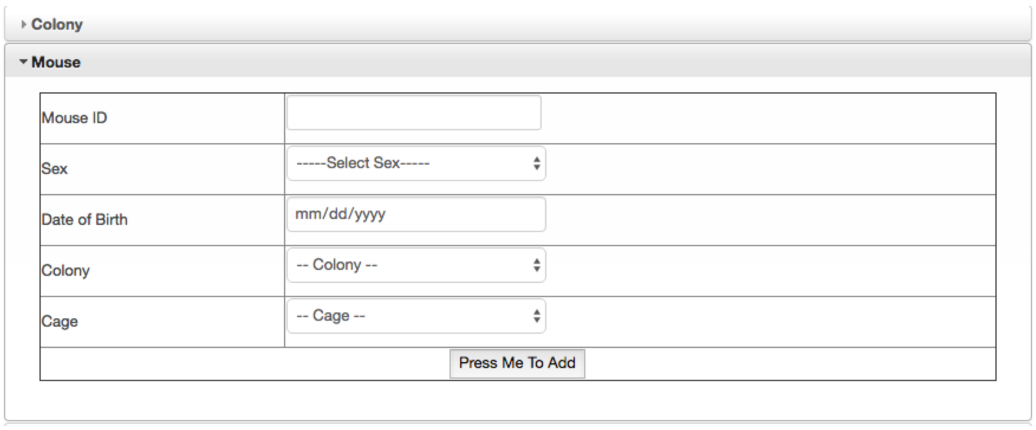
10.7. Data management

For this part, we have security constraint on the experiment settings. If people login with student, he/she will not update the experiment settings tables. If supervisor login, he/she can update and delete the experiment settings input. Supervisors have privilege to edit and delete genotype, phenotype, colony, treatment, surgery, supervisors table.



▾ Genotype

ID	Name	Description	Action	Delete
1	C57	C57	Edit Save	Delete
2	HCN_SOD1	HCN_SOD1	Edit Save	Delete
3	Serotonin	Serotonin	Edit Save	Delete
4	LCN_SOD1	LCN_SOD1	Edit Save	Delete
5	Control_HCN	Control_HCN	Edit Save	Delete
6	Control_LCN	Control_LCN	Edit Save	Delete
7	geno_name	23 22	Edit Save	Delete



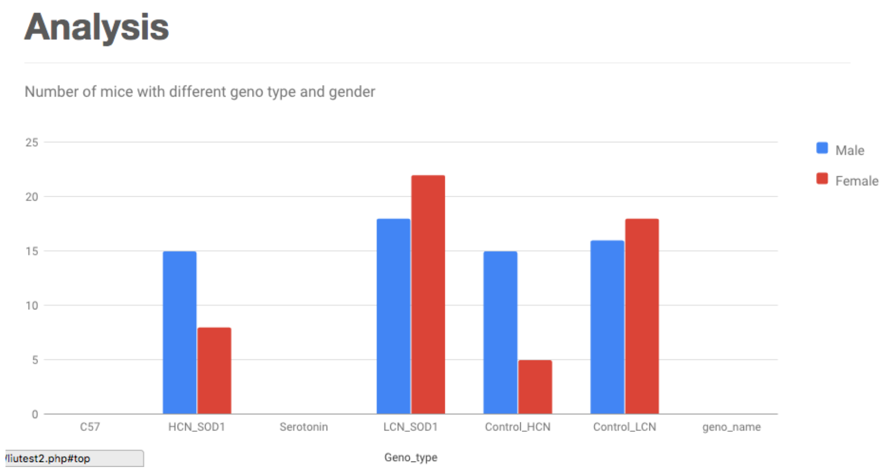
Mouse ID	
Sex	-----Select Sex-----
Date of Birth	mm/dd/yyyy
Colony	-- Colony --
Cage	-- Cage --

If students login, they will not update the tables.

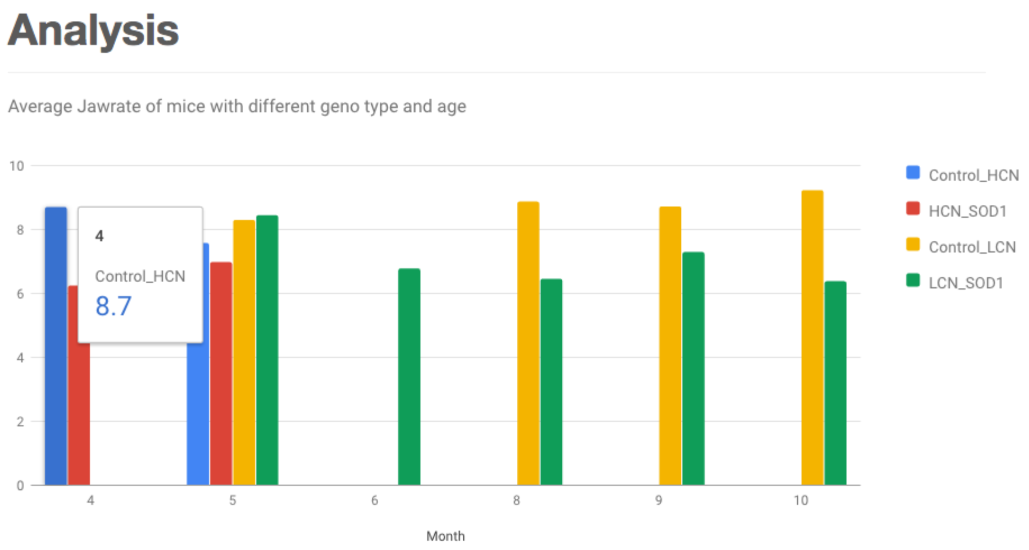
10.8. Analysis Functions:

We can do some analysis functions to visualize some distributions.

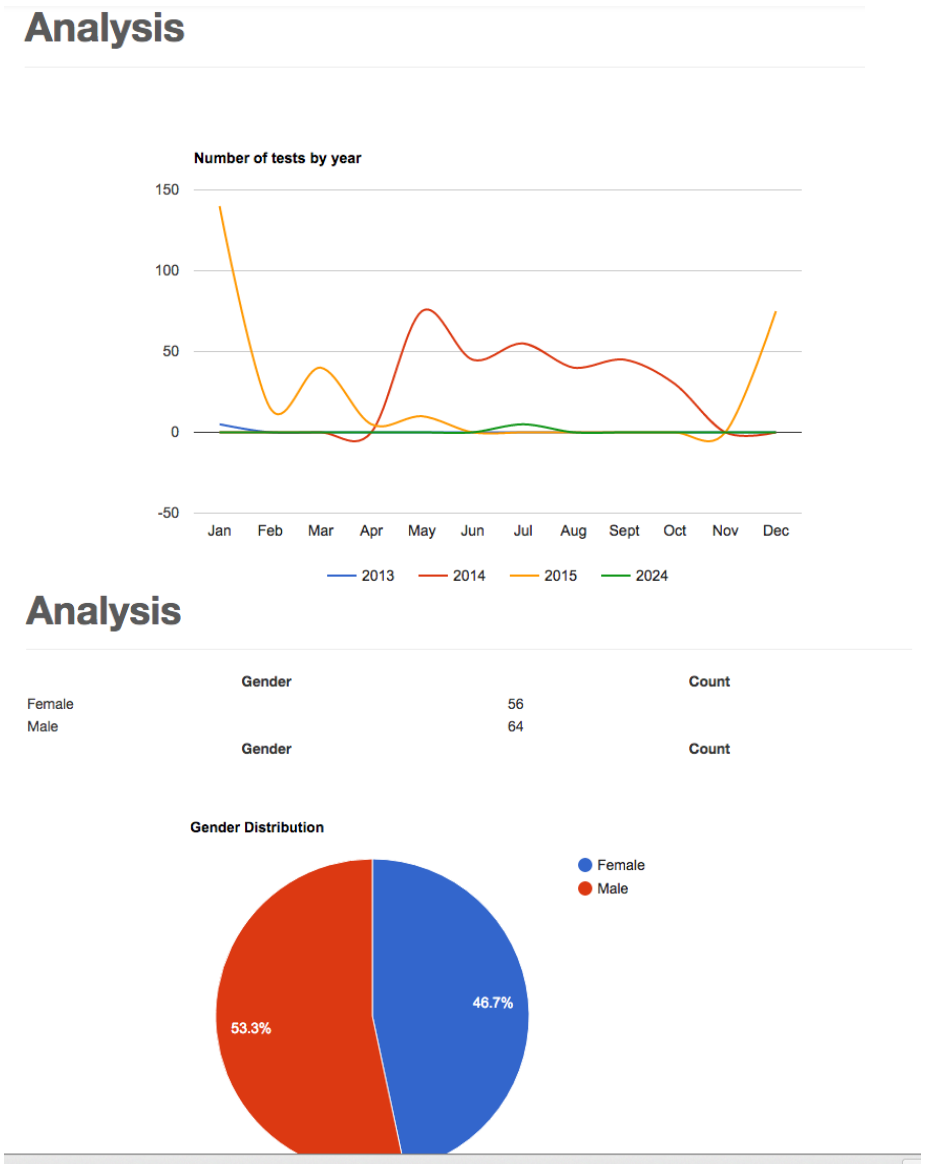
For each genotype, compare the male and female test objects:



For each group of age, analysis different type of genotype.



For each month


Analysis

Analysis

	Gender	Count
Female		56
Male		64
	Gender	Count

10.9. Log Out and send e-mail function

User can click the logout button on the top menu to logout and re-login.
By clicking the send email button, you can send an email to Teresa Lever.