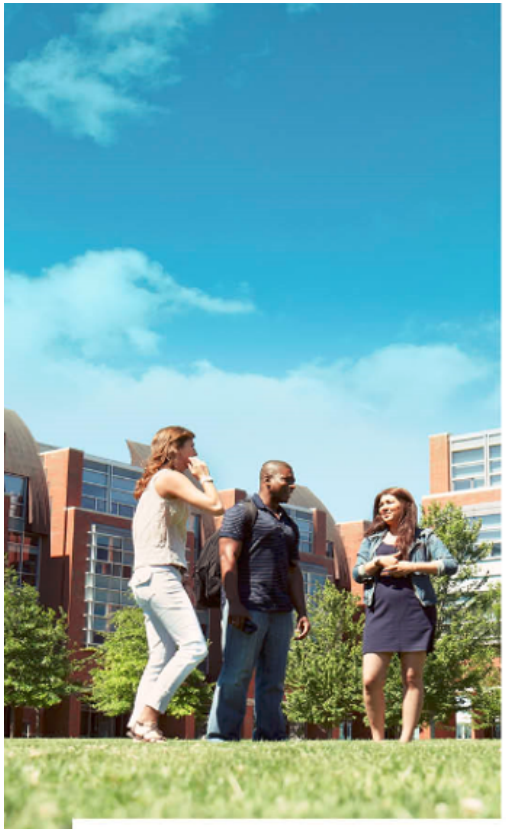


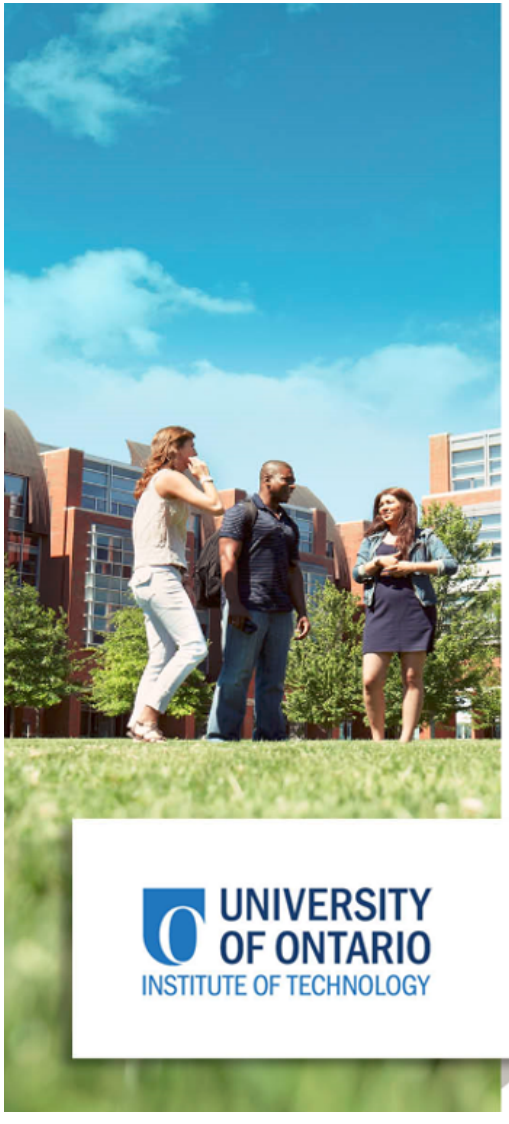
MONGODB AND MONGOOSE

A DOCUMENT DATABASE

Randy J. Fortier
randy.fortier@uoit.ca
@randy_fortier

OUTLINE

- MongoDB
  - The MongoDB Client
  - Mongoose
    - Creates (inserts)
    - Reads
    - Updates
    - Deletes



MONGODB AND MONGOOSE

MONGODB

MONGODB

- A (JSON) document database
  - Each document has a unique index, called _id
  - Documents can have any properties (dynamic)
  - Documents can be hierarchical
- MongoDB instances can share the load

RUNNING MONGODB

- Run the Mongo daemon
  - By default, the server (daemon) runs on port 27019

```
$ mongod
```

RUNNING MONGODB

- Run the client interface

```
$ mongo
>
```

MONGODB CLIENT OPERATIONS

- Create a new database (a group of collections):

```
> use mydb;
```

MONGODB CLIENT OPERATIONS

- Create a new database (a group of collections)
- Insert a new document into a collection:

```
> db.names.insertOne({firstName: 'Philip', lastName: 'Frye'});
> db.names.insertMany([{firstName: 'Taranga', lastName: 'Leela'},
                       {firstName: 'Bender', lastName: 'Rodriguez'}])
```

MONGODB CLIENT OPERATIONS

- Create a new database (a group of collections)
- Insert a new document into a collection
- Query a collection for matching documents:

```
> db.names.find().pretty();
> db.names.find({firstName: 'Philip'});
> db.names.find({$or: [{firstName: 'Philip'}, {firstName: 'Amy'}]);
> db.students.find({gpa: {$gt: 3.0}});
> db.students.find({program: {$regex: /.*Computer.*/}})
```

MONGODB CLIENT OPERATIONS

- Create a new database (a group of collections)
- Insert a new document into a collection
- Query a collection for matching documents
- Update a document in a collection:

```
> db.names.updateOne({firstName: 'Philip'}, {$set: {lastName: 'Fry'}}
> db.names.updateMany({firstName: 'Philip'}, {$set: {lastName: 'Fry'}
```

MONGODB CLIENT OPERATIONS

- Create a new database (a group of collections)
- Insert a new document into a collection
- Query a collection for matching documents
- Update a document in a collection
- Delete a document from a collection

```
> db.names.deleteOne({firstName: 'Philip'});
> db.names.deleteMany({firstName: 'Philip'});
```

MONGODB AND MONGOOSE

MONGOOSE

MONGOOSE

- An API for accessing MongoDB from Node.js/Express
  - The MongoDB client could be used directly
    - MongoDB's client is JavaScript
  - Mongoose provides object-relational mapping (ORM)
  - Operations:
    - create - insertOne
    - read - find
    - update - updateOne
    - delete - remove

SCHEMA

- A schema describes a document's structure
  - This schema is used to validate documents

```
var mongoose = require('mongoose');
mongoose.connect('localhost:27017/university');
var Schema = mongoose.Schema;
var studentSchema = new Schema({
        sid: {type: String,
         validate: [/^1[0-9]{8}$/, 'must be 9 digits'],
         unique: true,
         index: true},
        firstName: String,
        lastName: String,
        gpa: Number,
}, {collection: 'students'});
var Student = mongoose.model('student', studentSchema);
```

CRUD OPERATIONS

- Create (insert a new document into a collection)

```
var newStudent = new Student({firstName: 'Taranga', lastName: 'Leela'
newStudent.save(function(error) {
    if (error) {
        // insert failed
        console.log('error while adding student:', error);
        reloadStudentList(request, response, 'Unable to add student')
    } else {
        // insert successful
        reloadStudentList(request, response, 'Student added');
    }
});
```

CRUD OPERATIONS

- Create (insert a new document into a collection)
- Read (query for documents)

```
Student.find({sid: '100000001'}).then(function(results) {
    if (results.length > 0) {
        // found a match
    } else {
        // no match
    }
});
```

CRUD OPERATIONS

- Create (insert a new document into a collection)
- Read (query for documents)
- Update a document

```
Student.update({sid: sid}, {gpa: 4.3}, {multi: false}, function(error
    if ((error) || (numAffected.nModified != 1)) {
        // the student was not updated
    } else {
        // the student was updated
    }
});
```

CRUD OPERATIONS

- Create (insert a new document into a collection)
- Read (query for documents)
- Update a document

```
Student.findById(id, function(error, result) {
    if ((error) || (numAffected.nModified != 1)) {
        // the student was not found
    } else {
        result.gpa = 4.3;
        result.save(function(error, numAffected) {
            if ((error) || (numAffected.nModified != 1)) {
                // the student was not updated
            } else {
                // the student was updated
            }
        });
    }
});
```

CRUD OPERATIONS

- Create (insert a new document into a collection)
- Read (query for documents)
- Update a document
- Delete

```
Student.find({sid: sid}).remove(function(error, numAffected) {
    if ((error) || (numAffected.nModified != 1)) {
        // the student was not deleted
    } else {
        // the student was deleted
    }
});
```

WRAP-UP

- In this section, we learned about:
  - MongoDB
    - MongoDB Client
    - Mongoose