Installation instructions:
  1.) unzip <Trove.src.zip> into desired folder
  2.) Modify <portNum> of trove-dev/server/config/environment/index.js -- port: process.env.PORT || <portNum> to desired serving port.
  3.) Start up a mongod (MongoDB) instance
  4.) To deploy, type in the command `npm start`

Recommended Usage:
  Deploy TROVE in two instances.
    Instance 1 will be the main listener for HL7 JSON messages.
    Instance 2 will be the main web server.
    See step 2.) for assigning different ports to the two instances

Database Information (MongoDB):
  When `npm start` is called from 4.) of Installation instructions, it will create a database instance, 'trove-dev'. In trove dev, there are two main collections to be associated with:
    - Studies (db.studies)
    - Users (db.users)

  Users must be of the form:
```
{
  "alt_name" : "Zoe Miller",
  "badges" : [],
  "full_name" : "Zoe Anne Miller",
  "minnies" : 1766,
  "userId" : "118",
  "username" : "zam7001"
}
```

  The full_name is used with assistant_radiologist and radiologist to associate the order with a "userId". The username is used with passportjs to do authentication.

  IMPORTANT users must be created manually/or from script previously before any data can be stored from the HL7 JSON listener route

Authentication + User Creation:
  Please see:
    http://danialk.github.io/blog/2013/02/23/authentication-using-passportjs/ & https://github.com/jaredhanson/passport-local/blob/master/examples/express3-mongoose/app.js
  for information associated with authentication. The current distribution will require a developer to modify the code for authentication to work properly.

Server HL7 JSON processing Information:
  The main route for processing HL7 information is located in trove-dev/server/api/study/study.controller.js
    /processHL7JSON
    this method expects JSON messages of the form:
```
{
  'accession':<String>,
  // Assistant Radiologist names should be of form 'Doe, John'
  // If this field is not filled, the server will try to parse a name from the 'report'. The name is found by matching the string pattern : var regex = /.*Prepared\sBy:(.*?)\|/;
  'assistant_radiologist':<String>,
  // Radiologist names should be of form 'Doe, John'
  // If this field is not filled, the server will try to parse a name from the 'report'. The name is found by matching the string pattern: var regex = /.*Study\sinterpreted\sand\sreport\sapproved\sby:(.*?)\|/;
  'radiologist':<String>,
  'report':<String>,
  'service_code':<String>,
  'service_description':<String>,
  'result_time':<String>,
  // Result status general is a single character of form: 'P', 'F', ...
  // First 'P' status report is the report filed by Assistant Radiologist/Resident. Any 'P' status message recieved afterwards will be treated as a possibly edited report by the attending radiologist
  // 'F' status means the report has been finalized.
  'result_status':<String>,
  'scheduled_time':<String>, of form i.e. '201501010808'
    '/d/d/d/d' year '/d/d' month '/d/d' date '/d/d' hour '/d/d' minutes  '/d/d' seconds. In that order.
  'transcribed_time':<String>, of form i.e. '201501010808'
  'completed_time':<String> of form i.e. '201501010808'
}
```