DICOM in Dart (DCMiD)

Project 13

Damish Shah    Danielle Tinio

Mentor: Dr. James Philbin
Topic and Goal

Determine the feasibility of using binary DICOM for building browser based medical imaging applications

Method:

• Design and implement a DICOM editor that reads and writes binary DICOM and displays it using HTML5, CSS3 and the Dart programming language.

• Test performance by reading, displaying and writing DICOM studies in binary format.

• Goal: Read and display imaging studies in less than 3 seconds.
Dependencies

✓ Access to our mentor
✓ Computer to write code
✓ Bitbucket to share code
✓ Dart & DICOM Reference Information
✓ Access to DICOM Test Data
DICOM Review

- **Dataset**: list of Attributes
- **Attributes**: can be Sequence
- **Sequence**: list of Items

Other datatypes:
- Ex: stringList, decimalList, Bulkdata Reference

Each item contains
• define members whose body returns a single expression
  • bobLikes() => isDeepFried || (hasPieCrust && !vegan);

• ‘?’ can be used in place of “if-else” statements
  • a = condition ? b: c

• Function expressions
  • var names = people.map((person) => person.name);

• Underscores for private methods and variables
  • int _test;

• Getters and setters
  • int get test => this._test;
  • void set test (int value) {
      this._test = value;
  }

* Example code from https://www.dartlang.org/articles/style-guide/
Work To Date

• Our parsing and writing is functional
  • Binary parsers
  • String parsers
  • Data structure
  • Created classes
    • DateTime to override Dart’s DateTime class
      • Needed to write more accurate time
  • Write Output
• Validating parsers with testing
• Developing the basic skeleton of UI for end-point user
Example code

- Binary data is being stored as ByteData in our ByteBuffer class
- Bytedata has a lot of built-in functions for binary data types, int in general
  - `int getInt8`
  - `Int getUint32`

```dart
class bytebuf {
  _int8 readUint8() {
    var val = _bd.getInt8(_chkRdIdx(_rdIdx));
    _rdIdx += _int8Size;
    return val;
  }
}
```

* Example code from our bytebuf.dart class
  * _bd is the internal ByteData representation of our binary data.
Future

• Give values when it becomes available
• Do not have to parse in time with everything else
• Asynchronous model for functions doing potentially expensive work

```
static readFile(File file) {
  Future handler = file.readAsBytes();
  handler.then((Uint8List bytes) {
    return new ByteBuf.fromBytes(bytes);
  });
}
```

*Example code from our ByteBuffer.dart class:
Problems

• Updating our code outline as we learn more about Dart
• We have found better ways to structure our code and have been forced to redo pieces of it.
• Parsers have not been affected, but how we handle input and the underlying data structure has had to be rewritten.
• As a result, the tests have to be updated as the methods are reorganized and optimized
  • Complete validation of output can be formally done once the parsers are finalized using unit tests

```dart
void main() {
    test('Addition test', () {
        expect(2 + 2 == 4, isTrue);
    });
}
```

4 PASSED, 1 FAILED, 0 ERRORS
What we plan to do

- To continue toward our maximum deliverables, we chose to split the upcoming tasks
  - Optimize parsers (Damish)
  - Validate the most recent version of code (Both)
  - Finish the user interface (Danielle)

- Continue our current frequency of meetings
  - Monday and Thursday at 9:30 with our mentor
  - Sunday, Monday, Wednesday, Friday at 10:00 as a team
Deliverables

• Minimum deliverables (March 20) → (April 5)
  ✓ Read and display DICOM in a browser and then write it
  • Build a test program that compares input and output to validate correctness (in progress)
  • Create unit tests for each class (in progress)

• Expected deliverables (April 3) → (April 8)
  • Display a work list of studies of n patients (in progress)
  • Display patient as collapse/expand tree for study information model (in progress)

• Maximum deliverables (May 1)
  • Display images
  • Add overlay information (abandoned due to time)
  • Edit metadata
  • Encrypt and decrypt studies using AES (GCM) using an encryption framework created at Hopkins Security Institute → (Summer 2014)
## Updated Project Plan

- **February 20**: Have project proposal finished and all of the programming planned and reviewed by Dr. Philbin
- **March 6**: Read input (parse)
- **March 20 → April 5**: Write and validate output
- **April 3 → April 8**: HTML5/CSS3 display metadata
- **May 1**: Display images
- **May 9**: Final Poster Presentation

<table>
<thead>
<tr>
<th></th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Proposal</td>
<td>20</td>
<td>6</td>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td>Read input (parse)</td>
<td>27</td>
<td>13</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>Validate output</td>
<td></td>
<td>20</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Display metadata in browser</td>
<td></td>
<td>27</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Display images</td>
<td></td>
<td></td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>Final Presentation</td>
<td></td>
<td></td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>
Questions?