Task 3:
Create a terminology server for FHIR R4 version based upon the HAPI-FHIR Starter Project: https://github.com/hapifhir/hapi-fhir-jpaserver-starter

a. The terminology server should contain references for Conditions based upon ICD-10(https://icd.who.int/browse10/2019/en#/).

b. The terminology server should contain references for measurement units based upon LOINC codes (https://loinc.org/downloads/).

Answer:
a) For this we have generated a json with ICD-10 condition codes and posted it in Local HAPI server UI in the CodeSystem resource. We have taken "code" and "display" information from the ICD-10 url:https://icd.who.int/browse10/2019/en#/ and populated it in a json. Please find the json as : **ICD_10Conditions.json**.

b) For this we have generated a json with LOINC codes and posted it in Local HAPI server UI in the CodeSystem resource. We have downloaded the csv files with Loinc codes from the given URL: https://loinc.org/downloads/ and converted the codes, display and definition into JSON format. Please find the json as **Loinc.json.**

Task 7:
Explore and modify the Disease Modules (Condition), to generate ICD-10 conditions.

Answer:
In the existing project all the Conditions were based on SNOMED-CT codes. As a part of this task we have mapped it to ICD-10 condition codes. Please find the generated jsons in the synthea\src\main\resources\modules.

Task 8:
Enable Synthea to generate data in alternative geographic locations, such as Europe. Use a relevant geographical standard**.**

Answer:

For this task we have cloned https://github.com/synthetichealth/synthea-international.git project inside synthea (https://github.com/synthetichealth/synthea.git) folder. We have performed the following steps:

a) git clone https://github.com/synthetichealth/synthea
b) git clone https://github.com/synthetichealth/synthea-international
c) cd synthea-international
d) cp -R xx/* ../synthea (xx- country code for European countries)
e) cd ../synthea
f) ./run_synthea -p 5 Nordrhein-westfalen Aachen (Query to generate 5 patients from Nordrhein-westfalen Aachen)

Task 9:
Generate the following type of FHIR R4 resources in a flexible and configurable way:

Questionnaire Responses:
https://www.hl7.org/fhir/questionnaireresponse.html

Answer:

We have made changes in the synthea module for this task. Changes were tagged with //FIT PROJECT CHANGES

Please follow these steps to run this project:

a) Clone the project https://github.com/shilpa2503/synthea.git
b) cd synthea
c) ./gradlew build -x test
d) ./run_synthea -p 5 Nordrhein-westfalen Aachen - This command will generate data of five patients from germany and will also have the QuestionnaireResponse resource added.

Below are the code changes in the respective components:

resources\synthea.properties

```
#FIT PROJECT CHANGES
generate.providers.questionnaire.default_file =
providers/questionnaire.csv
```

src\main\java\org\mitre\synthea\engine\Generator.java

```
    //FIT PROJECT CHANGES
    // Initialize Questionnaire
    try{
      String fileName =
Config.get("generate.providers.questionnaire.default_file", "false");
      Provider.loadQuestionnaire(fileName);
    }
    catch(IOException e) {
        System.out.println(e);
    }
```

src\main\java\org\mitre\synthea\world\agents\Provider.java

```
//FIT PROJECT CHANGES
  private static ArrayList<Questionnaire> QuestionnaireList = new
ArrayList<Questionnaire>();
```

```
//FIT PROJECT CHANGES
  private static ArrayList<Questionnaire> QuestionnaireList = new
ArrayList<Questionnaire>();
   /**
   * Given a line of parsed CSV input, convert the data into a
Questionnaire.
   * @param line - read a csv line to a Questionnaire's attributes
   * @return A Questionnaire.
   */
  private static Questionnaire
csvLineToQuestionnaire(Map<String,String> line) {
    Questionnaire q = new Questionnaire();
    // using remove instead of get here so that we can iterate over
the remaining keys later
    q.id = line.remove("id");
    q.category = line.remove("category");
    q.subCategory = line.remove("Subcategory");
    q.items = line.remove("Items");
    q.scales = line.remove("Scales");
    return q;
  }

  public static void loadQuestionnaire(String filename)
      throws IOException {
    String resource = Utilities.readResource(filename);
    Iterator<? extends Map<String,String>> csv =
SimpleCSV.parseLineByLine(resource);

    while (csv.hasNext()) {
      Map<String,String> row = csv.next();
      Questionnaire parsed = csvLineToQuestionnaire(row);
      QuestionnaireList.add(parsed);
    }
  }

  public static ArrayList<Questionnaire> getQuestionnaireResponse() {
    return QuestionnaireList;
  }

}
```

src\main\java\org\mitre\synthea\world\agents\Questionnaire.java

```
package org.mitre.synthea.world.agents;

import java.io.Serializable;

import org.json.JSONException;
import org.json.JSONObject;

public class Questionnaire implements Serializable {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public String id;
    public String category;
    public String subCategory;
    public String items;
    public String scales;

    public String getJSONData() {

        JSONObject json = new JSONObject();
        try {

            json.put("category", category);
            json.put("subCategory", subCategory);
            json.put("items", items);
            json.put("scales", scales);
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return json.toString();
    }
}
```

src\main\java\org\mitre\synthea\export\FhirR4.java

```
//FIT PROJECT CHANGES
import
org.hl7.fhir.r4.model.QuestionnaireResponse.QuestionnaireResponseStat
us;
```

```
//FIT PROJECT CHANGES
import org.mitre.synthea.world.agents.Questionnaire;
```

```
//FIT PROJECT CHANGES
import org.hl7.fhir.r4.model.QuestionnaireResponse;
```

```
      //FIT PROJECT CHANGES
      for (Questionnaire questionResponse: encounter.responses) {
        questionnaireResponse(person, personEntry, bundle,
encounterEntry, questionResponse);
      }
```

```
  //FIT PROJECT CHANGES
    /**
   * Map the given Observation with attachment element to a FHIR
QUestionnaire resource, and add it to the
   * given Bundle.
   *
   * @param rand           Source of randomness to use when
generating ids etc
   * @param personEntry    The Entry for the Person
   * @param bundle         Bundle to add the Media to
   * @param encounterEntry Current Encounter entry
   * @param response   The Observation to map to FHIR and add to the
bundle
   * @return The added Entry
   */
  private static BundleEntryComponent
questionnaireResponse(RandomNumberGenerator rand,
          BundleEntryComponent personEntry, Bundle bundle,
BundleEntryComponent encounterEntry,
          Questionnaire response) {
    org.hl7.fhir.r4.model.QuestionnaireResponse questionnaireResource
=
        new org.hl7.fhir.r4.model.QuestionnaireResponse();



questionnaireResource.setStatus(QuestionnaireResponseStatus.INPROGRES
S);
    questionnaireResource.setEncounter(new
Reference(encounterEntry.getFullUrl()));
    questionnaireResource.setQuestionnaire(response.getJSONData());
    return newEntry(rand, bundle, questionnaireResource);
  }
```

src\main\java\org\mitre\synthea\world\concepts\HealthRecord.java

```
import java.io.IOException;
```

```
    /**
     * Java Serialization support for the prescriptionDetails field.
     * @param ois stream to read from
     */
    private void readObject(ObjectInputStream ois) throws
ClassNotFoundException, IOException {
      ois.defaultReadObject();
      String prescriptionJson = (String) ois.readObject();
      if (prescriptionJson != null) {
        Gson gson = Utilities.getGson();
        this.prescriptionDetails = gson.fromJson(prescriptionJson,
JsonObject.class);
```

```
            }
        }
    }
```