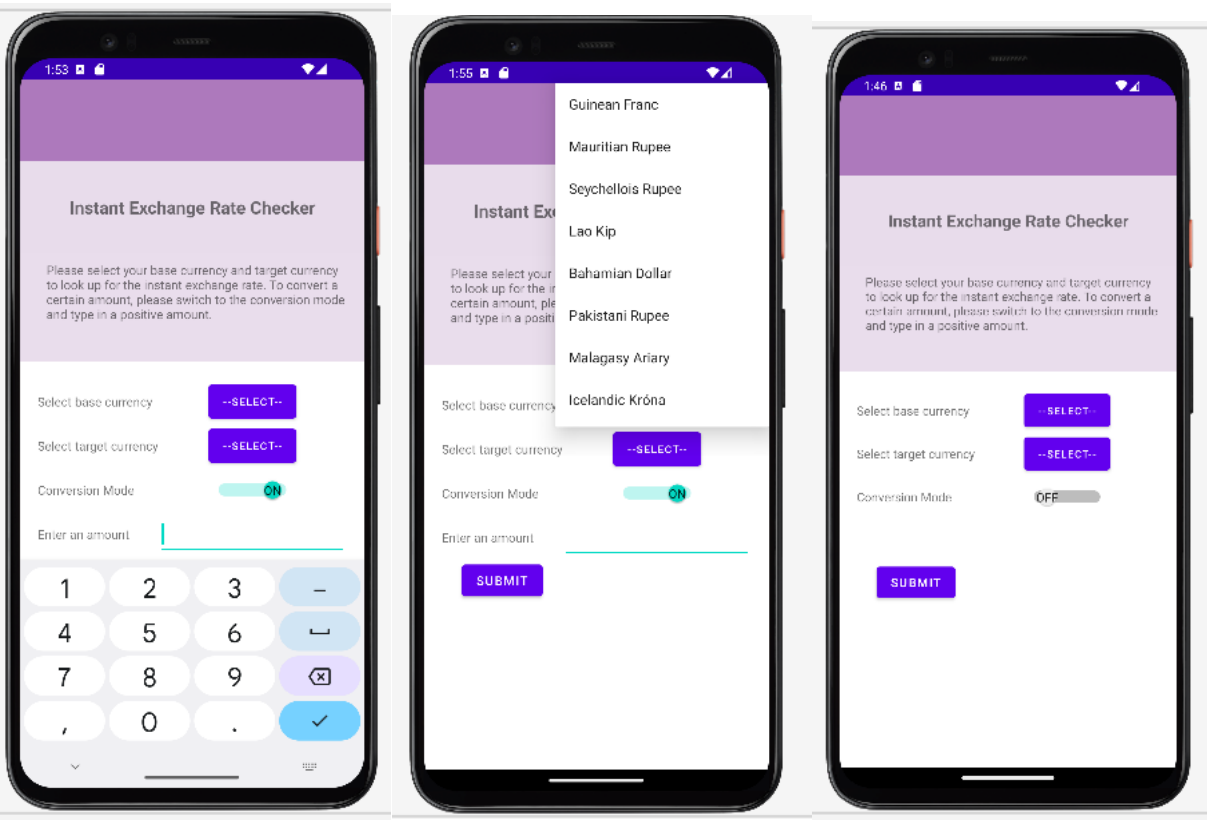
1.

The layout uses TextView, button, switch, and an EditText when Conversion Mode is on. When users click on the "—SELECT--" button, there will be a popup menu where users need to select from the menus before submit. If the conversion mode is on, users are also required to give a number that's greater than 0.



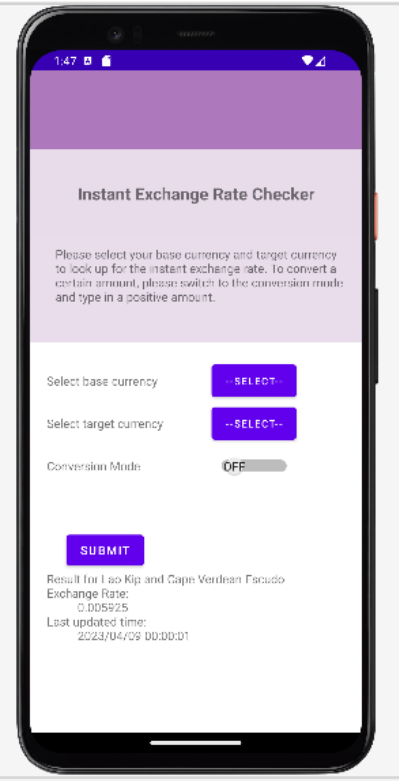
The submit button will send the http requests to the server as specified below.

```
try {
    // GET wants us to pass the message on the URL line
    URL url = new URL( spec: "https://wendy-wyt-vigilant-telegram-7x5xwg59xxwfwp5-8080.preview.app.github.dev/api/" + message);
//      URL url = new URL("http://10.0.2.2:8080/Project4Task2-1.0-SNAPSHOT/api/" + message);
    conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    // we are sending plain text
    conn.setRequestProperty("Content-Type", "text/plain; charset=utf-8");
    // tell the server what format we want back
    conn.setRequestProperty("Accept", "text/plain");
```

The request and the response will be parsed to or from JSON as the following codes specified.

```
    private Response search(Request request) {
        // prepare the message to server
        Gson gson=new Gson();
        String message= gson.toJson(request);
        Response httpResponse=null;
        // send messages to server and receive result from server
        String result= "";
        result = HttpConnector.doGet(message);
        // handle successful connection and results
        if(!result.equals("")){
            // parse the result message to a response object
            httpResponse=gson.fromJson(result,Response.class);
        }
        return httpResponse;
    }
}
```

New information will be displayed in a text view under the submit button as demonstrated below. Users can also repeat the process by select the currencies and other inputs again. It then click on the submit button to execute the action again.



2.

The web server will receive messages from the requests and then write the response back to users as the codes demonstrated.

```
no usages  new *
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String action=request.getServletPath(); // action of the request

    // prepare the appropriate DOCTYPE (mobile or website) for the view pages
    String ua = request.getHeader( s: "User-Agent");
    if (ua != null && ((ua.indexOf("Android") != -1) || (ua.indexOf("iPhone") != -1))) {...} else {...}
    String nextView="index.jsp";

    // call methods to retrieve results from third-party API
    if(action.contains("/api")){
        String requestMsg=request.getPathInfo()
                .replace( target: "/", replacement: ""); // request message with all necessary info
        String responseMsg=getExRate(requestMsg,response); // response message with answers
        // write message to the response and send to users
        response.getWriter().write(responseMsg);
        response.getWriter().flush();
    // call methods to regenerate the analytics
    }else if(action.contains("/dashboard Screenshot(Alt + A)
}
```

After receiving the message from the user, the message will be parsed to Request object, then the Request object will be used to form a url to the 3rd Party API. After getting the url from the API, the server will then parse it to an ExRate object with all information from the API. The server will further parse the ExRate object to a Response object with all information needed for response to the previous request. The Response object will be further parsed to JSON message and then sent to Android users.

```
1 usage  new *
public ExRate searchExRate(String base, String target){
    ExRate result=null;
    // read from url and receive the response in json
    String url="https://v6.exchangerate-api.com/v6/"+APIKEY+"/pair/"+base+"/"+target;
    String jsonStr=fetch(url);
    // if the request fails, return -1 error code
    if(Objects.equals(jsonStr, b: "")) return result;
    // convert the json string to json object
    Gson gson=new GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
    result=gson.fromJson(jsonStr, ExRate.class);
    result.parseDateTime();

    // record data to database
    recordJobData(result);
    return result;
}
```

4 & 5

After receiving the response from API, data will be stored to database. The data includes time_last_update_utc,base_code,target_code,conversion_rate, conversion_result, and the record time (hour, date of month, month).

```
    * Take the input with key-value pairs and write it to a new document
    * @param input Formatted input of content for document
    * @return 1 if the storage is successful; 0 if the storage fails
    * https://www.mongodb.com/docs/drivers/java/sync/v4.3/fundamentals/crud/query-document/
    */
1 usage  new *
public int insertDoc(String input){
    // find the target collection
    MongoCollection collection=null;
    try{...}catch(IllegalArgumentException e){...}

    Document doc=new Document(); // new document to be inserted
    // parse input to the document
    String pairs[]=input.replaceAll( regex: "[{}]", replacement: "").split( regex: ",");
    for(String pair: pairs){...}
    // add the record time to the document
    LocalDateTime now=LocalDateTime.now();
    doc.append("record_hour",now.getHour());
    doc.append("record_month_date",now.getDayOfMonth());
    doc.append("record_month",now.getMonthValue());

    // add the document to collection
    try{
        collection.insertOne(doc);
    }catch(MongoWriteException e){...}catch(MongoWriteConcernException e){...}catch(MongoException e){...}
    return 1;
}
```

The following codes set up the connection with MongoDB

```
1 usage  new *
private void setMongoDB(){
    ConnectionString connectionString = new ConnectionString("mongodb://yutingwu:LBIJ6VLPm7z1sA7b@" +
            "ac-itetvwv-shard-00-02.ggwtpnj.mongodb.net:27017," + // cluster url
            "ac-itetvwv-shard-00-01.ggwtpnj.mongodb.net:27017," + // cluster url
            "ac-itetvwv-shard-00-00.ggwtpnj.mongodb.net:27017" + // cluster url
            "/project4task1?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1");
    MongoClientSettings settings = MongoClientSettings.builder()
            .applyConnectionString(connectionString)
            .serverApi(ServerApi.builder()
                    .version(ServerApiVersion.V1)
                    .build())
            .build();
    MongoClient mongoClient = MongoClients.create(settings);
    database = mongoClient.getDatabase( s: "Project4Task1");
}
```

6.

Operations analytics

Operations Analytics

Top 5 searched currency

EUR(9)
USD(6)
ISK(5)
JPY(2)
LKR(1)

Peak time (hour, day of month, month)

Peak Hour of Day--Hour(Count)
22(4)
Peak Date of Month--Date(Count)
9(8)
Peak Month--Month(Count)
4(15)

Total search of today

Server has completed 8 requests

Full Logs from Database

_id	time_last_update_utc	base_code	target_code	conversion_rate	conversion_result	record_hour	record_month_date	record_month
$oid: 643204d47a80f61157b1faf7	2023/04/09 00:00:01	USD	EUR	0.9168	0.0	20	8	4
$oid: 643204d47a80f61157b1faf8	2023/04/09 00:00:01	USD	EUR	0.9168	0.0	20	8	4
$oid: 643204d47a80f61157b1faf9	2023/04/09 00:00:01	USD	EUR	0.9168	0.0	20	8	4
$oid: 6432267a0146502a4597657e	2023/04/09 00:00:01	EUR	ISK	149.6947	0.0	22	8	4
$oid: 643228120146502a4597657f	2023/04/09 00:00:01	USD	EUR	0.9168	0.0	22	8	4
$oid: 643228b9b878665f3642791c	2023/04/09 00:00:01	USD	EUR	0.9168	9.168	22	8	4
$oid: 643228f3b878665f3642791d	2023/04/09 00:00:01	SCR	ISK	9.8612	197.224	22	8	4
$oid: 6432370ef21ae8055debd566	2023/04/09 00:00:01	USD	EUR	0.9168	9.168	3	9	4
$oid: 64323731f21ae8055debd567	2023/04/09 00:00:01	JPY	EUR	0.006951	0.06951	3	9	4
$oid: 64323a42af975c440f7d423b	2023/04/09 00:00:01	JPY	EUR	0.006951	0.06951	4	9	4
$oid: 64323aa6af975c440f7d423c	2023/04/09 00:00:01	MUR	ISK	3.0193	0.0	4	9	4
$oid: 64323abdaf975c440f7d423d	2023/04/09 00:00:01	LKR	ISK	0.4263	8.526	4	9	4
$oid: 64324db9fbb42f5caa6d9f3f	2023/04/09 00:00:01	LAK	ISK	0.008061	0.0	1	9	4
$oid: 64324ddcfbb42f5caa6d9f40	2023/04/09 00:00:01	GNF	MGA	0.5112	5.112	1	9	4
$oid: 64324e90b64db76416987ff3	2023/04/09 00:00:01	LRD	XDR	0.004513	0.0	5	9	4

7.

To deploy the server to the codespace, I first created a ROOT.war and Dockerfile. In github codespace, I uploaded the files mentioned above. And then created a new `.devcontainer.json in codespace.` `And then I started it with the following commands:`

```
docker build -t interesting_picture .
docker images
docker run --rm -it -p 8080:8080 interesting_picture
```

(replace interesting_picture with project4task2)

And then I changed the port to visible in Ports tab.