G11 - B6214562 นายเฉลิมเกียรติ คงกะพัน

ระบบหลัก : ระบบประวัติผู้ป่วย

ระบบย่อย : ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab)

ระบบหลัก : G11 – ระบบประวัติผู้ป่วย

ระบบย่อย : ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab)

1. Requirements (เขียนบรรยาย + User story)

Requirements

ระบบประวัติผู้ป่วยของโรงพยาบาลแห่งหนึ่ง เป็นระบบที่ให้ผู้ใช้ระบบทำการกรอกข้อมูลประวัติผู้ป่วยเพื่อทำการลงทะเบียนเป็นเวชระเบียนผู้ป่วยในสังกัดของโรงพยาบาล ทำการเก็บข้อมูลส่วนตัว ซักประวัติและอาการเบื้องต้นของผู้ป่วยที่เข้ามารับการรักษาที่โรงพยาบาล ระบบเก็บข้อมูลจากห้องปฏิบัติการเป็นระบบที่นักเทคนิคการแพทย์จะทำการตรวจผู้ป่วยด้วยการส่งสิ่งตรวจของผู้ป่วยเข้าห้องปฏิบัติการ เช่น การตรวจผลเลือดหรือสิ่งอื่นๆที่ไม่สามารถตรวจจากภายนอกได้แยกตามแต่ละประเภทการตรวจ และผู้ใช้ระบบซึ่งเป็นนักเทคนิคการแพทย์ทำการตรวจและบันทึกผลการตรวจจากห้องปฏิบัติการลงในระบบ จากนั้นนักเทคนิคการแพทย์ยังสามารถส่งต่อผลการตรวจจากห้องปฏิบัติการที่สามารถบอกถึงข้อมูลสำคัญเพื่อให้แพทย์นำไปใช้ในการประกอบการวินิจฉัยและวิเคราะห์หาวิธีการรักษาโรคของผู้ป่วยต่อได้

User story (ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab))

ในบทบาทของ นักเทคนิคการแพทย์

ฉันต้องการ บันทึกผลการตรวจจากห้องปฏิบัติการ

เพื่อ เก็บข้อมูลผลการตรวจจากห้องปฏิบัติการเพื่อให้บุคลากรทางการแพทย์สามารถนำไปใช้ในการวิเคราะห์ ค้นหาโรคและวิธีการรักษาผู้ป่วย

Output หน้าจอ นักเทคนิคการแพทย์บันทึกผลการตรวจจากห้องปฏิบัติการ => ระบบบันทึกผลการตรวจจากห้องปฏิบัติการ

Output ข้อมูล ระบบบันทึกข้อมูลผลการตรวจจากห้องปฏิบัติการ

คำนามที่อาจจะกลายเป็น Entity(ตารางในฐานข้อมูล) ของระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab)

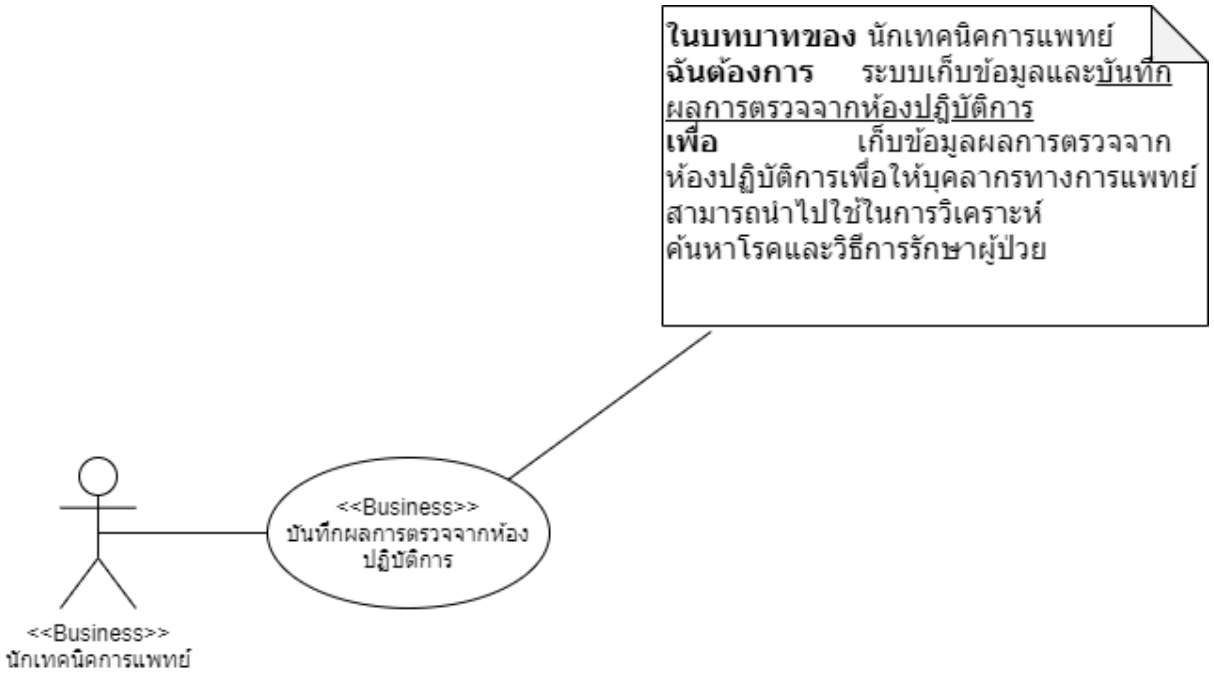
คำนาม	**เหตุผล :** เกี่ยวข้องกับ User Story หรือไม่
เวชระเบียน	เกี่ยวข้องโดยตรง เพราะเป็นการระบุตัวผู้ป่วย
นักเทคนิคการแพทย์	เกี่ยวข้องโดยตรง เพราะเป็นตัวข้อมูลบทบาท
ผลตรวจจากห้องปฏิบัติการ	เกี่ยวข้องโดยตรง เพราะต้องเก็บข้อมูลผลการตรวจจากห้องปฏิบัติการ
ห้องปฏิบัติการ	เกี่ยวข้องโดยตรง เพราะทำการบันทึกข้อมูลห้องปฏิบัติการที่ใช้ตรวจ
ประเภทการตรวจ	เกี่ยวข้องโดยตรง เพราะต้องเก็บข้อมูลประเภทการตรวจ

G11 - B6214562 นายเฉลิมเกียรติ คงกะพัน

ระบบหลัก : ระบบประวัติผู้ป่วย

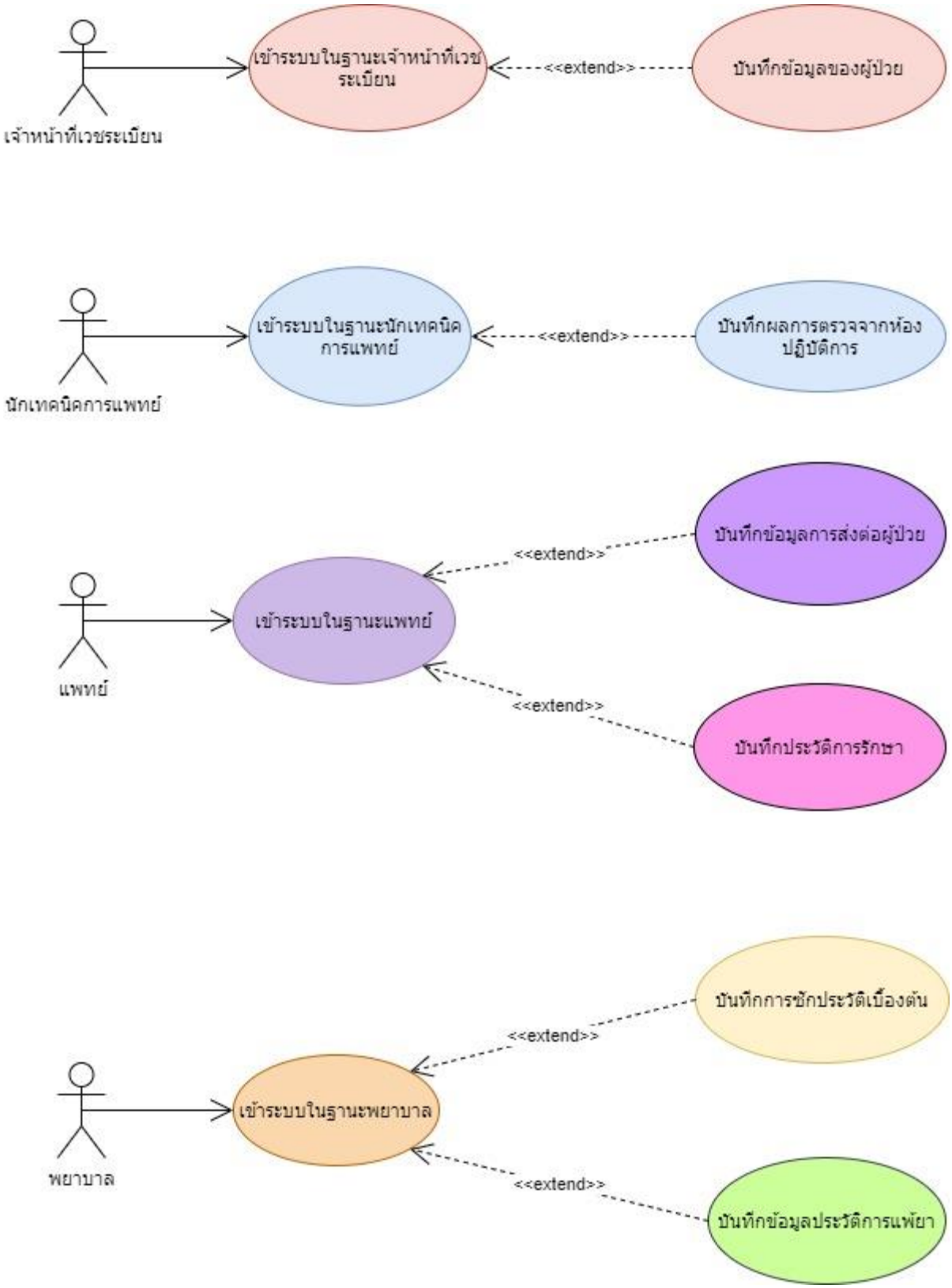
ระบบย่อย : ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab)

2. Business Use Case

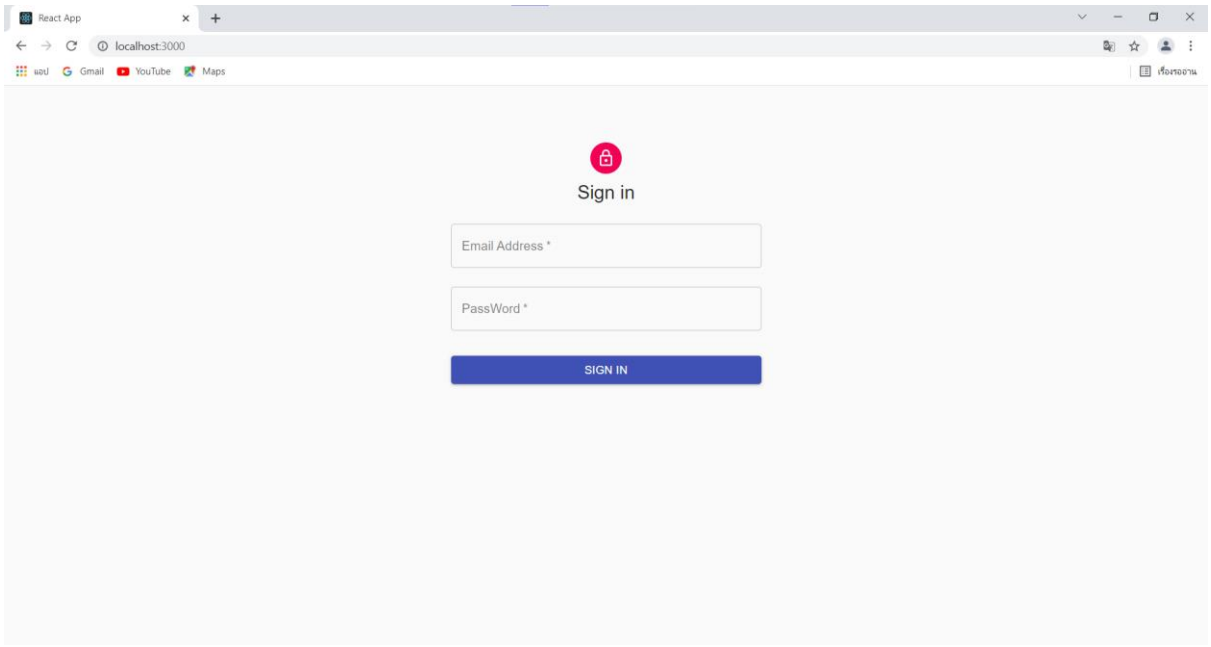


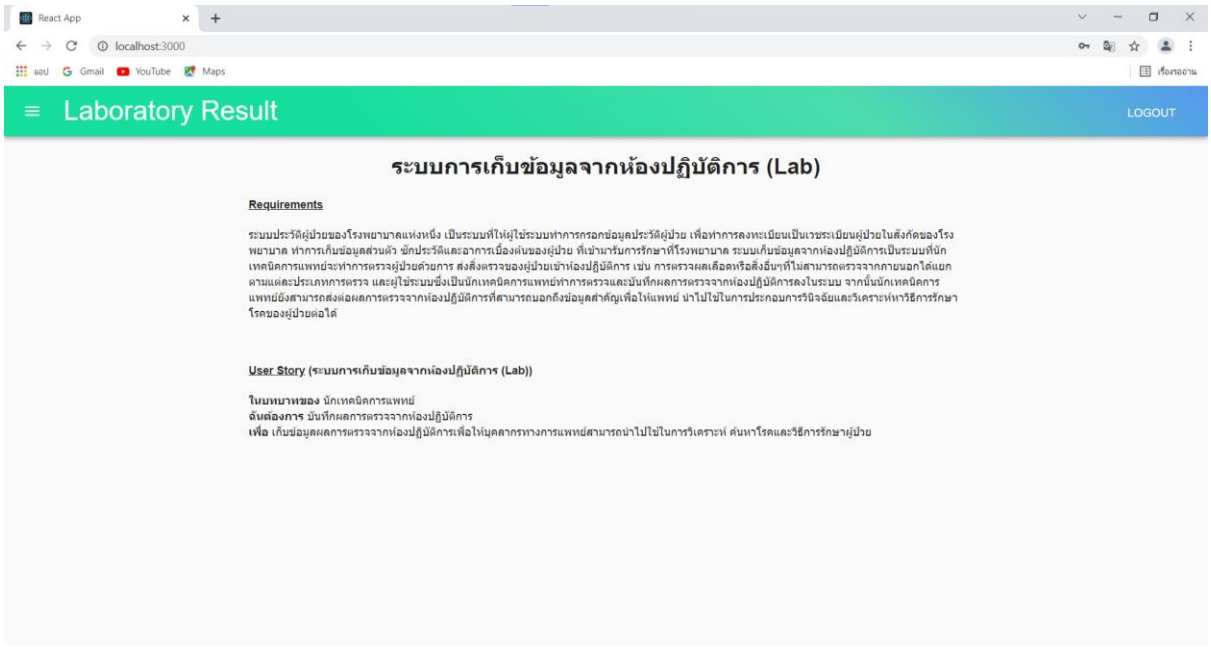
3. System Use Case

4. System Use Case รวมทั้งระบบใหญ่ ในหน้าเดียวกัน



5. User Interface

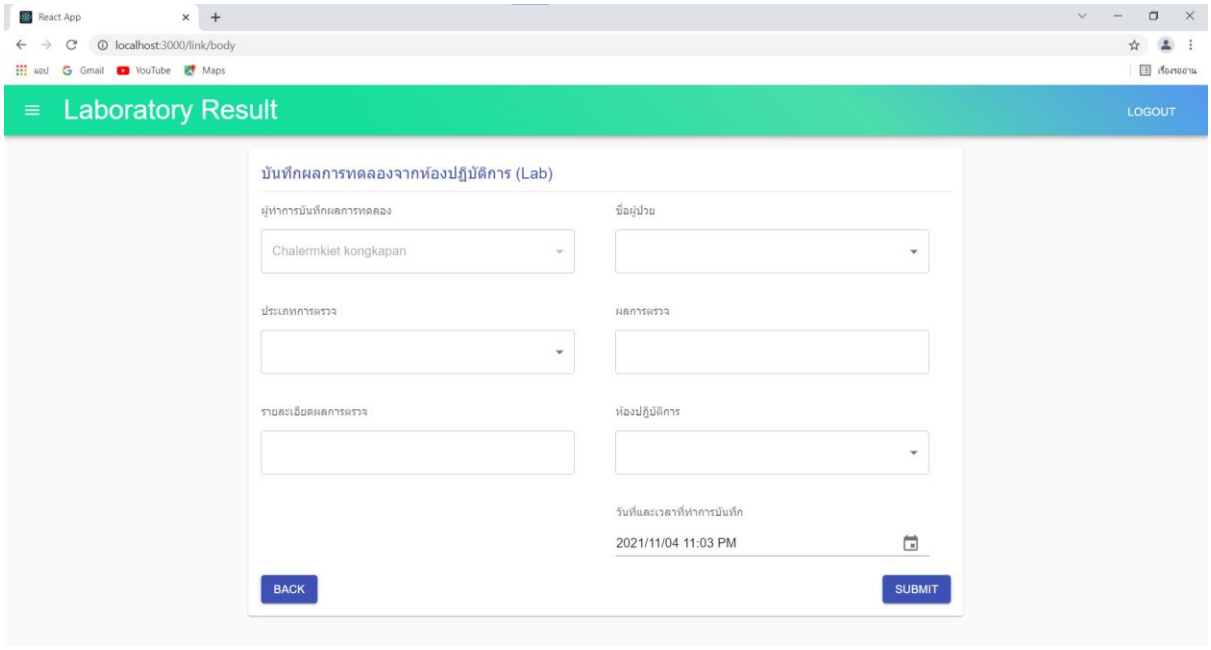


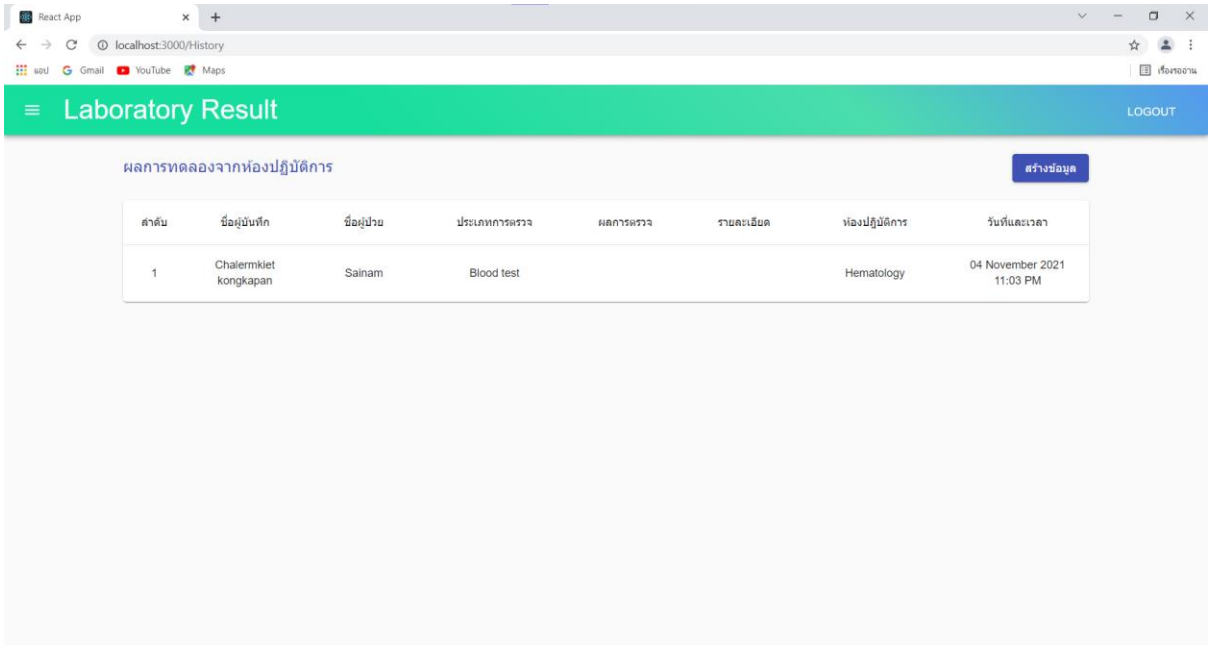


G11 - B6214562 นายเฉลิมเกียรติ คงกะพัน

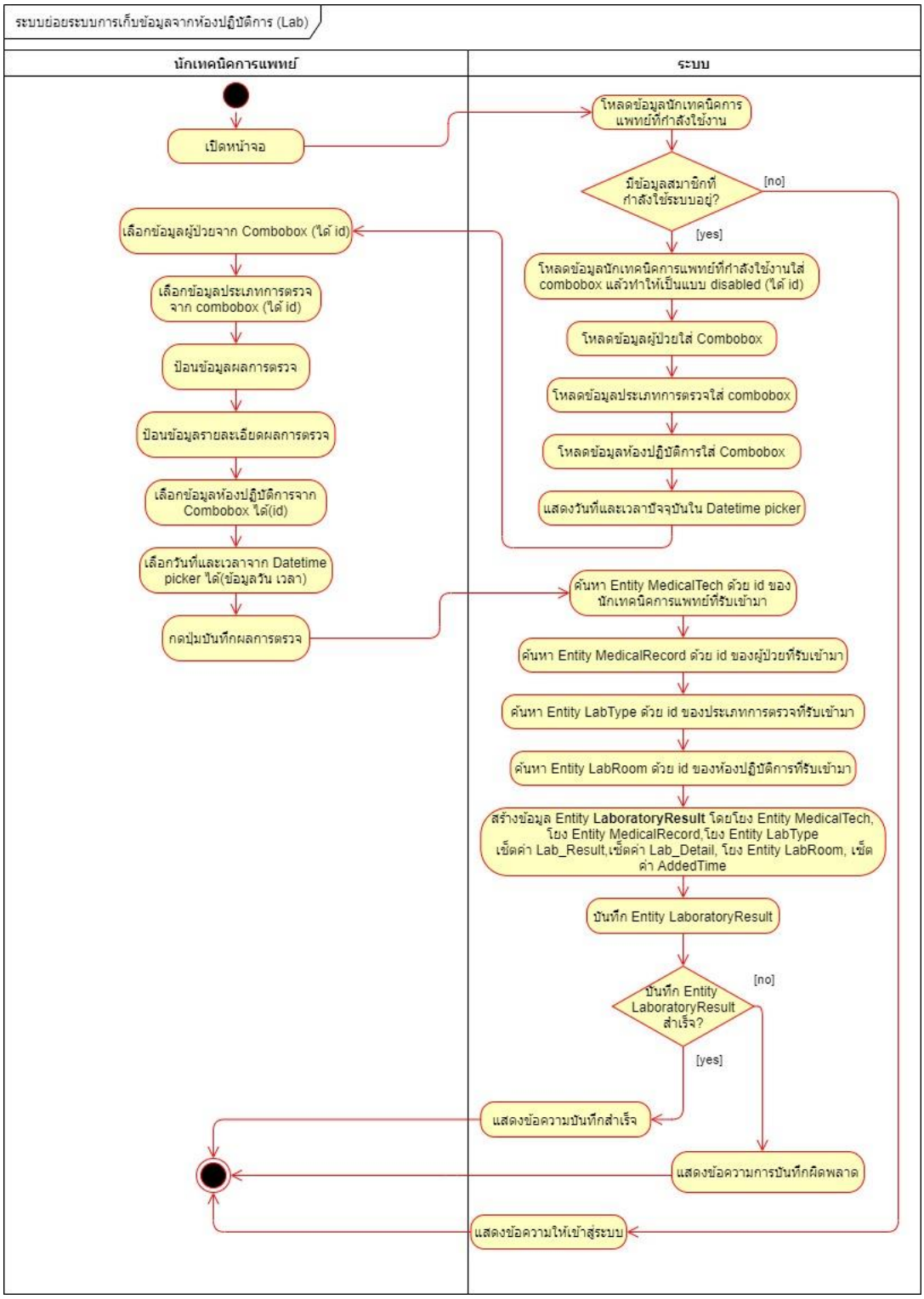
ระบบหลัก : ระบบประวัติผู้ป่วย

ระบบย่อย : ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab)

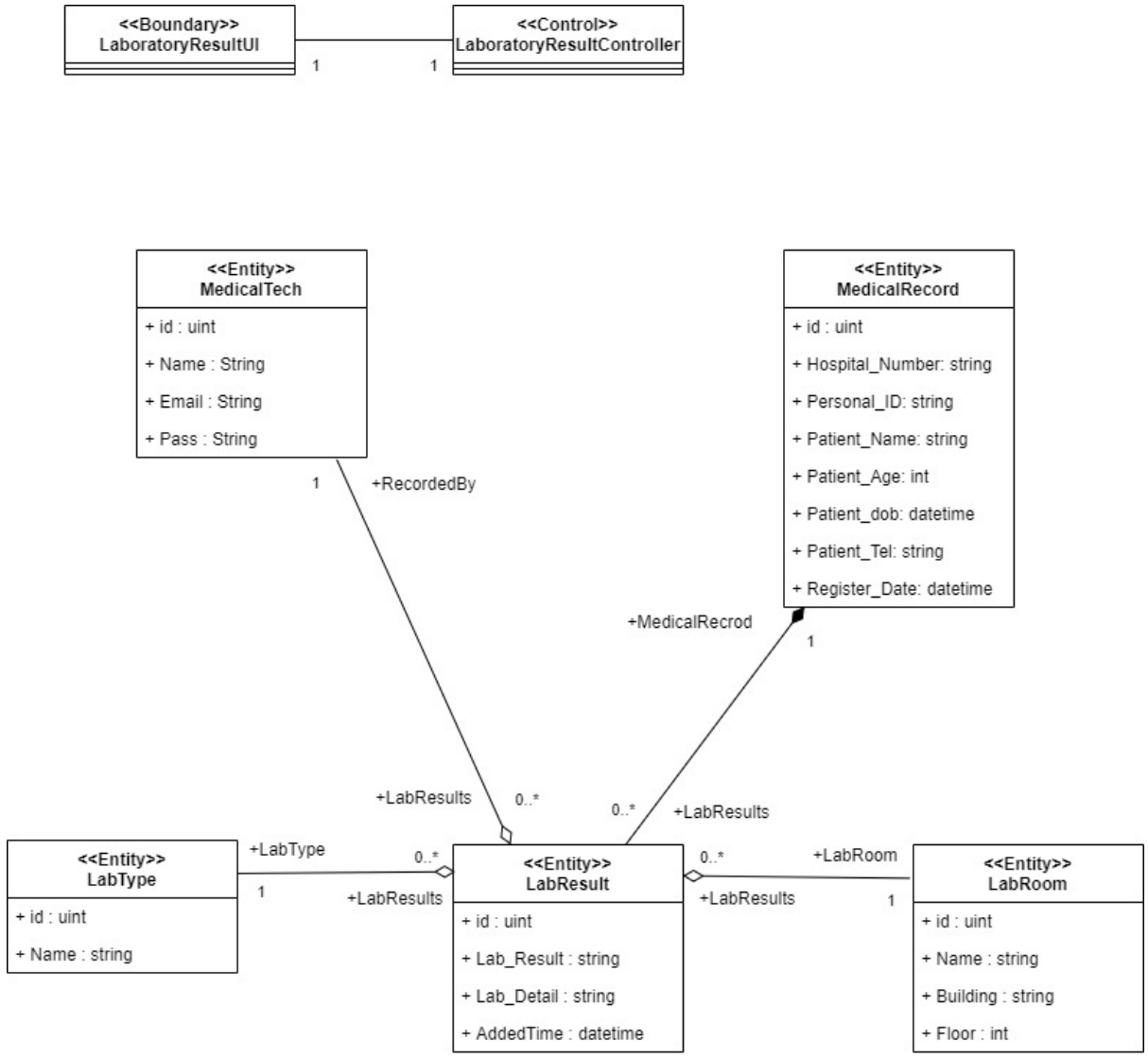




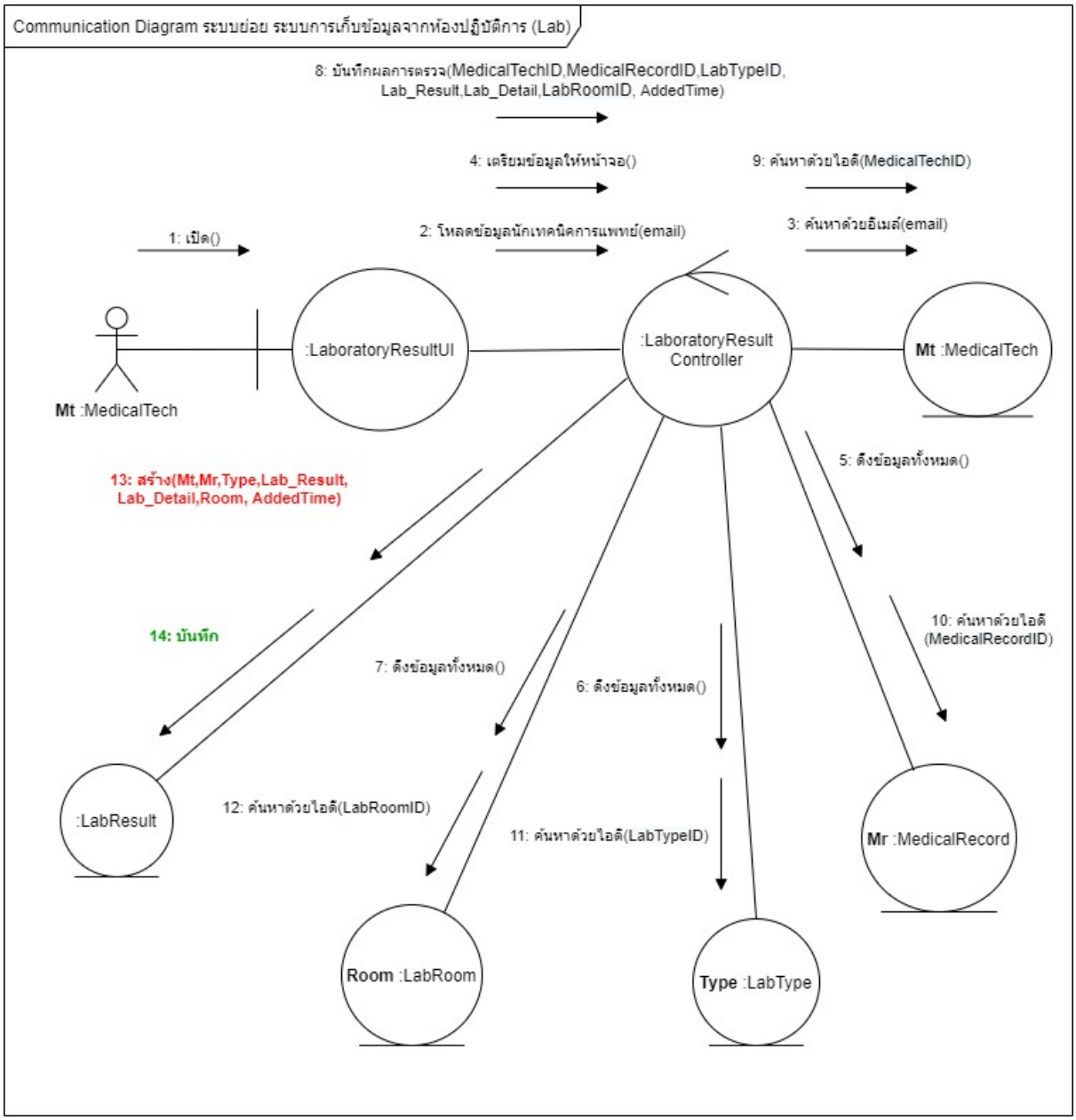
6. System Activity Diagram



7. Class Diagram ระดับ Design



8. Communication Diagram



9. Source Code

Source ของ React

ไฟล์ App.tsx

```
import React, { Fragment, useEffect, useState } from 'react';
import Body from "./components/Body";
import Navbar from "./components/Navbar";
import History from "./components/History";
import SignIn from "./components/SignIn";
import { BrowserRouter as Router, Switch, Route } from "react-router-
dom";
import Home from './components/Home';


function App() {
  const [token, setToken] = useState<string>("");

  useEffect(() => {
    const getToken = localStorage.getItem("token");
    if (getToken) {
      setToken(getToken);
    }
  }, []);

  if (!token) {
    return <SignIn />
  }

  return (
    <div>
      <Router>
        {token && (
          <Fragment>
            <Navbar/>
            <Switch>
              <Route exact path="/" component={Home} />
              <Route exact path="/History" component={History} />
              <Route exact path="/link/body" component={Body} />
            </Switch>
          </Fragment>
        )}
      </Router>
    </div>
  );
}
export default App;
```

ไฟล์ Components/Body.tsx

```
import React, { ChangeEvent,
    useEffect,
    useState,
    Fragment,
    SyntheticEvent } from 'react';
import { Link as RouterLink } from "react-router-dom";
import { makeStyles, Theme, createStyles } from "@material-
ui/core/styles";
import Typography from '@material-ui/core/Typography';
import Container from '@material-ui/core/Container';
import { Box, Paper } from "@material-ui/core";
import Divider from "@material-ui/core/Divider";
import Grid from '@material-ui/core/Grid';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import { Select } from '@material-ui/core';
import { MenuItem } from '@material-ui/core';
import { Snackbar } from '@material-ui/core';
import { Alert } from '@material-ui/lab';
import { FormControl } from '@material-ui/core';
import DateFnsUtils from '@date-io/date-fns';
import { MuiPickersUtilsProvider } from '@material-ui/pickers';
import { KeyboardDateTimePicker } from '@material-ui/pickers';
import { MedicalRecordInterface ,
        LabresultInterface,
        LabRoomInterface,
        LabTypeInterface,
        MedicalTechInterface} from "../model/LabResultUI";

const useStyles = makeStyles((theme: Theme) =>

    createStyles({

        root: { flexGrow: 1 },

        container: { marginTop: theme.spacing(2) },

        paper: { padding: theme.spacing(2), color:
theme.palette.text.secondary },

        table: { minWidth: 20 },

        textfield: { width: 400, },

        datefield: {
            marginLeft: theme.spacing(1),
            marginRight: theme.spacing(1),
            width: 200,
```

```
        },

    })
);

export default function Body() {
    useEffect(() => {
        getMedicalRecord();
        getLabRoom();
        getLabType();
      }, []);

    const classes = useStyles();
    const [Labresult,setLabresult] =
useState<Partial<LabresultInterface>>({});
    const MedicalTech: MedicalTechInterface =
(JSON.parse(localStorage.getItem("MedicalTech")|| ""));

    const  [AddedTime,setAddedTime] = useState<Date|null>(new Date());
    const handleAddedTime = (date: Date | null) => {
      setAddedTime(date);
    }


    const [MedicalRecord, setMedicalRecord] =
useState<MedicalRecordInterface[]>([]);
    const getMedicalRecord = async() => {
        const apiUrl = "http://localhost:8080/api/ListMedicalRecord";
        const requestOptions = {
          method: "GET",
          headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",},
        }

        fetch(apiUrl, requestOptions)
          .then((response) => response.json())
          .then((res) => {
            console.log(res.data);
            if(res.data) {
              setMedicalRecord(res.data)
            } else {
              console.log("else")
            }
          });
      }

      const [LabRoom, setLabRoom] = useState<LabRoomInterface[]>([]);
      const getLabRoom = async() => {
          const apiUrl = "http://localhost:8080/api/ListLabRoom";
```

```
          const requestOptions = {
            method: "GET",
            headers: {
              Authorization: `Bearer ${localStorage.getItem("token")}`,
              "Content-Type": "application/json",},
          }

          fetch(apiUrl, requestOptions)
            .then((response) => response.json())
            .then((res) => {
              console.log(res.data);
              if(res.data) {
                setLabRoom(res.data)
              } else {
                console.log("else")
              }
            });
        }

        const [LabType, setLabType] = useState<LabTypeInterface[]>([]);
        const getLabType = async() => {
          const apiUrl = "http://localhost:8080/api/ListLabType";
          const requestOptions = {
            method: "GET",
            headers: {
              Authorization: `Bearer ${localStorage.getItem("token")}`,
            "Content-Type": "application/json",},
          }

          fetch(apiUrl, requestOptions)
            .then((response) => response.json())
            .then((res) => {
              console.log(res.data);
              if(res.data) {
                setLabType(res.data)
              } else {
                console.log("else")
              }
            });
        }

      const handleLabresultChange = (event: ChangeEvent<{name?: string;
value: unknown}>) => {
        const name = event.target.name as keyof typeof Labresult;
        setLabresult({...Labresult, [name]: event.target.value,});
      };

      /* --- SUBMIT --- */
  const [success, setSuccess] = useState(false);
  const [error, setError] = useState(false);
```

```
  const handleClose = (event?: SyntheticEvent, reason?: string) => {
    if (reason === "clickaway") {
      return;
    }
    setSuccess(false);
    setError(false);
  };
  const submitLabresult = () => {
    let data = {
          MedicalTechID:  MedicalTech?.ID,
      MedicalRecordID:    Labresult.MedicalRecordID,
      LabTypeID:  Labresult.LabTypeID,
      Lab_Result: Labresult.Lab_Result,
          Lab_Detail: Labresult.Lab_Detail,
          LabRoomID:  Labresult.LabRoomID,
          AddedTime:  AddedTime
    };

    const apiUrl = "http://localhost:8080/api/CreateLabResult";
    const requestOptionsPost = {
      method: "POST",
      headers: {
        Authorization: `Bearer ${localStorage.getItem("token")}`,
        "Content-Type": "application/json",},
      body: JSON.stringify(data),
    };

    fetch(apiUrl, requestOptionsPost)
      .then((response) => response.json())
      .then((res) => {
        if (res.data) {
          setSuccess(true);
        } else {
          setError(true);
        }
    });
  }

    return (
        <Container className={classes.container} maxWidth="md">
          <Snackbar open={success} autoHideDuration={1000}
onClose={handleClose}
TransitionProps={{onExit:()=>(window.location.href="/History")}}>
            <Alert onClose={handleClose} severity="success">
      บันทึกข้อมูลสำเร็จ
            </Alert>
          </Snackbar>
          <Snackbar open={error} autoHideDuration={1000}
onClose={handleClose}>
            <Alert onClose={handleClose} severity="error">
```

```
บันทึกข้อมูลไม่สำเร็จ
          </Alert>
        </Snackbar>
          <Paper className={classes.paper}>
              <Box display="flex">
                  <Box flexGrow={1}>
                      <Typography
                          component="h2"

                          variant="h6"

                          color="primary"

                          gutterBottom
                      >
          บันทึกผลการทดลองจากห้องปฏิบัติการ (Lab)
                      </Typography>

                  </Box>
              </Box>
              <Divider />

              <Grid container spacing={3} className={classes.root}>


                  <Grid item xs={6}>
                      <p>ผู้ทำการบันทึกผลการทดลอง</p>
                      <Select variant="outlined"
                          disabled
                          defaultValue={0}
                          style={{ width: 400 }}
                      >
                          <MenuItem
value={0}>{MedicalTech.Name}</MenuItem>
                      </Select>
                  </Grid>

                  <Grid item xs={6}>
                      <p>ชื่อผู้ป่วย</p>
                      <Select variant="outlined"
                          value={Labresult.MedicalRecordID}
                          inputProps={{name: "MedicalRecordID"}}
                          onChange={handleLabresultChange}
                          style={{ width: 400 }}
                      >
                          <MenuItem value={0} key={0}>เลือกชื่อผู้ป่วย
</MenuItem>
                          {MedicalRecord.map((item:
MedicalRecordInterface) => (
```

```
                                <MenuItem value={item.ID}
key={item.ID}>{item.Patient_Name}</MenuItem>))}
                            </Select>
                        </Grid>
                        <Grid item xs={6}>
                            <p>ประเภทการตรวจ</p>
                            <Select variant="outlined"
                                value={Labresult.LabTypeID}
                                inputProps={{name: "LabTypeID"}}
                                onChange={handleLabresultChange}
                                style={{ width: 400 }}
                            >
                                <MenuItem value={0} key={0}>เลือกห้องปฏิบัติการ
</MenuItem>
                                {LabType.map((item: LabTypeInterface) => (
                                <MenuItem value={item.ID}
key={item.ID}>{item.Name}</MenuItem>))}
                            </Select>
                        </Grid>

                        <Grid item xs={6}>
                            <p>ผลการตรวจ</p>
                            <TextField
                                id="Lab_Result"
                                type="string"
                                inputProps={{name:"Lab_Result"}}
                                value={Labresult.Lab_Result || ""}
                                onChange={handleLabresultChange}
                                className={classes.textfield}
                                variant="outlined" />
                        </Grid>

                        <Grid item xs={6}>
                            <p>รายละเอียดผลการตรวจ</p>
                            <TextField
                                id="Lab_Detail"
                                type="string"
                                inputProps={{name:"Lab_Detail"}}
                                value={Labresult.Lab_Detail || ""}
                                onChange={handleLabresultChange}
                                className={classes.textfield}
                                variant="outlined" />
                        </Grid>

                        <Grid item xs={6}>
                            <p>ห้องปฏิบัติการ</p>
                                <Select variant="outlined"
                                    value={Labresult.LabRoomID}
                                    inputProps={{name: "LabRoomID"}}
                                    onChange={handleLabresultChange}
```

```
                                style={{ width: 400 }}
                            >
                                <MenuItem value={0} key={0}>เลือกห้องปฏิบัติการ
</MenuItem>
                                {LabRoom.map((item: LabRoomInterface) =>
(
                                <MenuItem value={item.ID}
key={item.ID}>{item.Name}</MenuItem>))}
                            </Select>
                    </Grid>


                    <Grid item xs={12}>
                        <FormControl style={{float:
"right",width:400,marginRight:27 }} variant="outlined">
                          <p>วันที่และเวลาที่ทำการบันทึก</p>
                            <MuiPickersUtilsProvider
utils={DateFnsUtils}>
                            <KeyboardDateTimePicker
                              name="WatchedTime"
                              value={AddedTime}
                              onChange={handleAddedTime}
                              minDate={new Date("2018-01-01T00:00")}
                              format="yyyy/MM/dd hh:mm a"
                            />
                            </MuiPickersUtilsProvider>
                          </FormControl>
                    </Grid>
                    <Grid item xs={6}>
                        <Button
                                variant="contained"
                                color="primary"
                                component={RouterLink}
                                to="/"
                                >BACK</Button>
                    </Grid>

                    <Grid item xs={6}>
                        <Button style={{ float: "right" }}
                                variant="contained"
                                color="primary"
                                onClick={submitLabresult}

                                >SUBMIT</Button>
                    </Grid>
                </Grid>
            </Paper>
        </Container>
    )
}
```

ไฟล์ Components/History.tsx

```
import React, { Fragment, useEffect, useState } from "react";
import { Link as RouterLink } from "react-router-dom";
import { createStyles, makeStyles, Theme } from "@material-
ui/core/styles";
import Typography from "@material-ui/core/Typography";
import Button from "@material-ui/core/Button";
import Paper from "@material-ui/core/Paper";
import Box from "@material-ui/core/Box";
import Table from "@material-ui/core/Table";
import TableBody from "@material-ui/core/TableBody";
import TableCell from "@material-ui/core/TableCell";
import TableContainer from "@material-ui/core/TableContainer";
import TableHead from "@material-ui/core/TableHead";
import TableRow from "@material-ui/core/TableRow";
import moment from 'moment';
import Container from '@material-ui/core/Container';
import IconButton from '@material-ui/core/IconButton';
import KeyboardArrowDownIcon from '@material-
ui/icons/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@material-ui/icons/KeyboardArrowUp';
import Collapse from '@material-ui/core/Collapse';
import { format } from 'date-fns'

import { LabresultInterface } from "../model/LabResultUI";

const useStyles = makeStyles((theme: Theme) =>
  createStyles({
    container: {marginTop: theme.spacing(3)},
    paper: {padding: theme.spacing(3)},
    table: {minWidth: 650},
    tableSpace: {marginTop: 20},
    row: {'& > !': {borderBottom: 'unset'},},
  })
);

export default function History() {
    const classes = useStyles();
    const [LabResult, setLabResult] =
useState<LabresultInterface[]>([]);

    const getLabResult = async() => {
      const apiUrl = "http://localhost:8080/api/ListLabResult";
      const requestOptions = {
        method: "GET",
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
          "Content-Type": "application/json",},
      };
```

```
      fetch(apiUrl, requestOptions)
        .then((response) => response.json())
        .then((res) => {
          console.log(res.data);
          if (res.data) {
            setLabResult(res.data);
          } else {
            console.log("else");
          }
        });
    };

    useEffect(() => {
      getLabResult();
    }, []);

    return (
      <div>
      <Container className={classes.container} maxWidth="lg">
        <Box display="flex">
          <Box flexGrow={1}>
            <Typography
              component="h2"
              variant="h6"
              color="primary"
              gutterBottom
            >
      ผลการทดลองจากห้องปฏิบัติการ
            </Typography>
          </Box>
          <Box>
            <Button
              component={RouterLink}
              to="/link/body"
              variant="contained"
              color="primary"
            >
      สร้างข้อมูล
            </Button>
          </Box>
        </Box>
        <TableContainer component={Paper}
className={classes.tableSpace}>
          <Table className={classes.table} aria-label="simple table">
            <TableHead>
              <TableRow>
                <TableCell align="center" width="5%">
        ลำดับ
                </TableCell>
```

```
                    <TableCell align="center" width="10%">
        ชื่อผู้บันทึก
                    </TableCell>
                    <TableCell align="center" width="10%">
        ชื่อผู้ป่วย
                    </TableCell>
                    <TableCell align="center" width="12%">
        ประเภทการตรวจ
                    </TableCell>
                    <TableCell align="center" width="10%">
        ผลการตรวจ
                    </TableCell>
                    <TableCell align="center" width="10%">
        รายละเอียด
                    </TableCell>
                    <TableCell align="center" width="12%">
        ห้องปฏิบัติการ
                    </TableCell>
                    <TableCell align="center" width="12%">
        วันที่และเวลา
                    </TableCell>
                  </TableRow>
                </TableHead>
                <TableBody>
                  {LabResult.map((item: LabresultInterface) => (
                    <TableRow key={item.ID}>
                      <TableCell align="center">{item.ID}</TableCell>
                      <TableCell
align="center">{item.MedicalTech.Name}</TableCell>
                      <TableCell
align="center">{item.MedicalRecord.Patient_Name}</TableCell>
                      <TableCell
align="center">{item.LabType.Name}</TableCell>
                      <TableCell
align="center">{item.Lab_Result}</TableCell>
                      <TableCell
align="center">{item.Lab_Detail}</TableCell>
                      <TableCell
align="center">{item.LabRoom.Name}</TableCell>
                      <TableCell align="center">{format((new
Date(item.AddedTime)), 'dd MMMM yyyy hh:mm a')}</TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
            </TableContainer>
          </Container>
        </div>
          )
        }
```

ไฟล์ Conponents/Home.tsx

```
import { createStyles, makeStyles, Theme } from "@material-
ui/core/styles";
import Container from "@material-ui/core/Container";
const useStyles = makeStyles((theme: Theme) =>
  createStyles({
    container: {
      marginTop: theme.spacing(2),
    },
    table: {
      minWidth: 650,
    },
    tableSpace: {
      marginTop: 20,
    },
  })
);

function Home() {
  const classes = useStyles();
  return (
    <div>
      <Container className={classes.container} maxWidth="md">
        <h1 style={{ textAlign: "center" }}>ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab)</h1>
        <h4><u>Requirements</u></h4>
        <p>
    ระบบประวัติผู้ป่วยของโรงพยาบาลแห่งหนึ่ง เป็นระบบที่ให้ผู้ใช้ระบบทำการกรอกข้อมูลประวัติผู้ป่วย
    เพื่อทำการลงทะเบียนเป็นเวชระเบียนผู้ป่วยในสังกัดของโรงพยาบาล ทำการเก็บข้อมูลส่วนตัว ซักประวัติและอาการเบื้องต้นของผู้ป่วย
    ที่เข้ามารับการรักษาที่โรงพยาบาล ระบบเก็บข้อมูลจากห้องปฏิบัติการเป็นระบบที่นักเทคนิคการแพทย์จะทำการตรวจผู้ป่วยด้วยการ
    ส่งสิ่งตรวจของผู้ป่วยเข้าห้องปฏิบัติการ เช่น การตรวจผลเลือดหรือสิ่งอื่นๆที่ไม่สามารถตรวจจากภายนอกได้แยกตามแต่ละประเภทการตรวจ
    และผู้ใช้ระบบซึ่งเป็นนักเทคนิคการแพทย์ทำการตรวจและบันทึกผลการตรวจจากห้องปฏิบัติการลงในระบบ
    จากนั้นนักเทคนิคการแพทย์ยังสามารถส่งต่อผลการตรวจจากห้องปฏิบัติการที่สามารถบอกถึงข้อมูลสำคัญเพื่อให้แพทย์
    นำไปใช้ในการประกอบการวินิจฉัยและวิเคราะห์หาวิธีการรักษาโรคของผู้ป่วยต่อได้
        </p>
        <br />
        <h4><u>User Story</u> (ระบบการเก็บข้อมูลจากห้องปฏิบัติการ (Lab))</h4>
        <p>
          <b>ในบทบาทของ</b>        นักเทคนิคการแพทย์<br />
          <b>ฉันต้องการ</b>  บันทึกผลการตรวจจากห้องปฏิบัติการ<br />
          <b>เพื่อ</b>       เก็บข้อมูลผลการตรวจจากห้องปฏิบัติการเพื่อให้บุคลากรทางการแพทย์สามารถนำไปใช้ในการวิเคราะห์
ค้นหาโรคและวิธีการรักษาผู้ป่วย<br />
        </p>
      </Container>
    </div>
  );
}
export default Home;
```

ไฟล์ Component/signIn.tsx

```
import React, { useState } from "react";
import Avatar from "@material-ui/core/Avatar";
import Button from "@material-ui/core/Button";
import CssBaseline from "@material-ui/core/CssBaseline";
import TextField from "@material-ui/core/TextField";
import LockOutlinedIcon from "@material-ui/icons/LockOutlined";
import Typography from "@material-ui/core/Typography";
import Snackbar from "@material-ui/core/Snackbar";
import MuiAlert, { AlertProps } from "@material-ui/lab/Alert";
import { makeStyles } from "@material-ui/core/styles";
import Container from "@material-ui/core/Container";
import { SigninInterface } from "../model/ISignin";

function Alert(props: AlertProps) {
  return <MuiAlert elevation={6} variant="filled" {...props} />;
}
const useStyles = makeStyles((theme) => ({
  paper: {
    marginTop: theme.spacing(8),
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
  },
  avatar: {
    margin: theme.spacing(1),
    backgroundColor: theme.palette.secondary.main,
  },
  form: {
    width: "100%",
    marginTop: theme.spacing(1),
  },
  submit: {
    margin: theme.spacing(3, 0, 2),
  },
}));

function SignIn() {
  const classes = useStyles();
  const [signin, setSignin] = useState<Partial<SigninInterface>>({});
  const [success, setSuccess] = useState(false);
  const [error, setError] = useState(false);

  const login = () => {
    const apiUrl = "http://localhost:8080/api/LoginMedicalTech";
    const requestOptions = {
      method: "POST",
      headers: { "Content-Type": "application/json" },
```

```
      body: JSON.stringify(signin),
    };
    fetch(apiUrl, requestOptions)
  .then((response) => response.json())
  .then((res) => {
        if (res.data) {
          setSuccess(true);
          localStorage.setItem("token", res.data.token);

localStorage.setItem("MedicalTech",JSON.stringify(res.data.medicaltech))
;
          window.location.reload()
        } else {
          setError(true);
        }
      });
  };

  const handleInputChange = (
    event: React.ChangeEvent<{ id?: string; value: any }>
) => {
    const id = event.target.id as keyof typeof signin;
    const { value } = event.target;
    setSignin({ ...signin, [id]: value });
  };

  const handleClose = (event?: React.SyntheticEvent, reason?: string) =>
{
    if (reason === "clickaway") {
      return;
    }
    setSuccess(false);
    setError(false);
  };

  return (
    <Container component="main" maxWidth="xs">
      <Snackbar open={success} autoHideDuration={6000}
onClose={handleClose}>
        <Alert onClose={handleClose} severity="success">
    เข้าสู่ระบบสำเร็จ
        </Alert>
      </Snackbar>
      <Snackbar open={error} autoHideDuration={6000}
onClose={handleClose}>
        <Alert onClose={handleClose} severity="error">
    อีเมลหรือรหัสผ่านไม่ถูกต้อง
        </Alert>
      </Snackbar>
```

```
      <CssBaseline />
      <div className={classes.paper}>
        <Avatar className={classes.avatar}>
          <LockOutlinedIcon />
        </Avatar>
        <Typography component="h1" variant="h5">
          Sign in
        </Typography>
        <form className={classes.form} noValidate>
          <TextField
            variant="outlined"
            margin="normal"
            required
            fullWidth
            id="Email"
            label="Email Address"
            name="Email"
            autoComplete="email"
            autoFocus
            value={signin.Email || ""}
            onChange={handleInputChange}
    />
          <TextField
            variant="outlined"
            margin="normal"
            required
            fullWidth
            name="Pass"
            label="PassWord"
            type="password"
            id="Pass"
            autoComplete="current-pass"
            value={signin.Pass || ""}
            onChange={handleInputChange}
    />
          <Button
            fullWidth
            variant="contained"
            color="primary"
            className={classes.submit}
            onClick={login}
          >
            Sign In
          </Button>
        </form>
      </div>
    </Container>
  );
}
export default SignIn;
```

ไฟล์ Components/Navbar.tsx

```
import React, { useState, useEffect } from "react";
import { Link as RouterLink } from "react-router-dom";
import { createStyles, makeStyles, Theme } from '@material-
ui/core/styles';
import AppBar from '@material-ui/core/AppBar';
import Button from "@material-ui/core/Button";
import Toolbar from '@material-ui/core/Toolbar';
import Typography from '@material-ui/core/Typography';
import MenuBookIcon from  "@material-ui/icons/MenuBook";
import AssignmentIcon from "@material-ui/icons/Assignment";
import Drawer from "@material-ui/core/Drawer";
import Divider from "@material-ui/core/Divider";
import IconButton  from "@material-ui/core/IconButton";
import List from "@material-ui/core/List"
import ListItem from "@material-ui/core/ListItem";
import ListItemIcon from "@material-ui/core/ListItemIcon";
import ListItemText from "@material-ui/core/ListItemText";
import MenuIcon from "@material-ui/icons/Menu";
import HomeIcon from "@material-ui/icons/Home";
import ExitToAppIcon from '@material-ui/icons/ExitToApp';

const useStyles = makeStyles((theme: Theme) =>
  createStyles({
    root: {
      flexGrow: 1,

    },
    title: {
      flexGrow: 1,
    },
    small: {
        marginTop : theme.spacing(0.5),
        marginRight : theme.spacing(2),
        width: theme.spacing(5),
        height: theme.spacing(5),

      },
    leftmagin:{
        marginLeft:theme.spacing(3),
    },
    colorbar:{
        background: 'linear-gradient(45deg, #16DE9D 30%, #4BDDAD 70%,
#5698F0 100%)',

    },
    menuButton: { marginRight: theme.spacing(2) },
    list: { width: 250 },
  }),
```

```
);

export default function ButtonAppBar() {
  const classes = useStyles();
  const SignOut = () => {
    localStorage.clear();
    window.location.href = "/";
  }
  const menu = [
    { name: "ผลการทดลอง", icon: <AssignmentIcon  />, path: "/History" },
    { name: "บันทึกผลการทดลอง", icon: <MenuBookIcon  />, path: "/link/body" },
  ]
  const [openDrawer, setOpenDrawer] = useState(false);
  const toggleDrawer = (state: boolean) => (event: any) => {
    if (event.type === "keydown" && (event.key === "Tab" || event.key
=== "Shift")) {
      return;
    }
    setOpenDrawer(state);
  }

  return (

    <div className={classes.root} >
      <AppBar position="static"className={classes.colorbar} >
        <Toolbar>
        <IconButton
            onClick={toggleDrawer(true)}
            edge="start"
            className={classes.menuButton}
            color="inherit"
            aria-label="menu"
          >
            <MenuIcon />
          </IconButton>
          <Drawer open={openDrawer} onClose={toggleDrawer(false)}>
            <List
              className={classes.list}
              onClick={toggleDrawer(false)}
              onKeyDown={toggleDrawer(false)}
            >
              <ListItem button component={RouterLink} to="/">
                <ListItemIcon><HomeIcon /></ListItemIcon>
                <ListItemText>หน้าแรก</ListItemText>
              </ListItem>
              <Divider />
              {menu.map((item, index) => (
                <ListItem key={index} button component={RouterLink}
to={item.path}>
                  <ListItemIcon>{item.icon}</ListItemIcon>
```

```
                  <ListItemText>{item.name}</ListItemText>
                </ListItem>
              ))}
              <ListItem button onClick={SignOut}>
              <ListItemIcon> <ExitToAppIcon/></ListItemIcon>
              <ListItemText>SignOut</ListItemText>
              </ListItem>

            </List>
          </Drawer>

          <Typography variant="h4"  className={classes.title}>
            Laboratory Result
          </Typography>
          <Button onClick={SignOut} color="inherit"
className={classes.small} >Logout</Button>

        </Toolbar>

      </AppBar>
    </div>
  );
}
```

ไฟล์ Models/ISignin.tsx

```
export interface SigninInterface {
    Email: string,
    Pass: string,
  }
```

ไฟล์ Models/LabResultUI.tsx

```
export interface MedicalRecordInterface {
    ID: number,
    Hospital_Number:    string,
      Personal_ID:    string,
      Patient_Name:     string,
      Patient_Age:      number,
      Patient_Dob: Date,
      Patient_Tel:      string,
      Register_Date: Date,
}
export interface LabresultInterface {
    ID: number,
      MedicalTechID:  number,
      MedicalTech: MedicalTechInterface,
    MedicalRecordID:    number,
      MedicalRecord: MedicalRecordInterface,
    LabTypeID:  number,
      LabType: LabTypeInterface
    Lab_Result: string,
      Lab_Detail: string,
      LabRoomID:  number,
      LabRoom: LabRoomInterface,
      AddedTime:  Date
}
export interface LabRoomInterface {
    ID: number,
      Name: string,
      Building:   string,
      floor:  number
}
export interface LabTypeInterface {
    ID: number,
      Name: string,
}
export interface MedicalTechInterface {
    ID: number,
      Name: string,
      Email:      string,
      Pass:  string
}
```

Source ของ controller เขียนด้วย Gin / Golang

ไฟล์ Controller/Actor/MedicalTech.go

```
package controller

import (
	"net/http"

	"github.com/Project/entity"
	"github.com/Project/service"
	"github.com/gin-gonic/gin"
	"golang.org/x/crypto/bcrypt"
)

// LoginMedicalTechPayload login body
type LoginMedicalTechPayload struct {
	Email string `json:"email"`
	Pass  string `json:"pass"`
}

// LoginMedicalTechResponse token response
type LoginMedicalTechResponse struct {
	Token       string             `json:"token"`
	MedicalTech entity.MedicalTech `json:"medicaltech"`
}

// POST /loginMedicalTech
func LoginMedicalTech(c *gin.Context) {
	var payload LoginMedicalTechPayload
	var MedicalTech entity.MedicalTech

	if err := c.ShouldBindJSON(&payload); err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	// ค้นหา MedicalTech ด้วย email ที่ผู้ใช้กรอกเข้ามา
	if err := entity.DB().Raw("SELECT * FROM medical_teches WHERE
email = ?", payload.Email).Scan(&MedicalTech).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

	// ตรวจสอบรหัสผ่าน
	err := bcrypt.CompareHashAndPassword([]byte(MedicalTech.Pass),
[]byte(payload.Pass))
	if err != nil {
```

```
            c.JSON(http.StatusBadRequest, gin.H{"error": "invalid
medical_teches credentials"})
            return
      }

      // กำหนดค่า SecretKey, Issuer และระยะเวลาหมดอายุของ Token สามารถกำหนดเองได้
      // SecretKey ใช้สำหรับการ sign ข้อความเพื่อบอกว่าข้อความมาจากตัวเราแน่นอน
      // Issuer เป็น unique id ที่เอาไว้ระบุตัว client
      // ExpirationHours เป็นเวลาหมดอายุของ token

      jwtWrapper := service.JwtWrapper{
            SecretKey:       "SvNQpBN8y3qlVrsGAYYWoJJk56LtzFHx",
            Issuer:          "AuthService",
            ExpirationHours: 24,
      }

      signedToken, err := jwtWrapper.GenerateToken(MedicalTech.Email)
      if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": "error signing
token"})
            return
      }

      tokenResponse := LoginMedicalTechResponse{
            Token:       signedToken,
            MedicalTech: MedicalTech,
      }

      c.JSON(http.StatusOK, gin.H{"data": tokenResponse})
}
```

ไฟล์ Controller/LabResult/LabRoom.go

```
package controller

import (
	"net/http"
	"github.com/Project/entity"
	"github.com/gin-gonic/gin"
)

func ListLabRoom(c *gin.Context) {
	var LabRoom []entity.LabRoom
	if err := entity.DB().Table("lab_rooms").Find(&LabRoom).Error; err
!= nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	c.JSON(http.StatusOK, gin.H{"data": LabRoom})
}
```

ไฟล์ Controller/LabResult/LabType.go

```
package controller

import (
	"net/http"
	"github.com/Project/entity"
	"github.com/gin-gonic/gin"
)

func ListLabType(c *gin.Context) {
	var LabType []entity.LabType
	if err := entity.DB().Table("lab_types").Find(&LabType).Error; err
!= nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	c.JSON(http.StatusOK, gin.H{"data": LabType})
}
```

ไฟล์ Controller/LabResult/LabResult.go

```
package controller

import (
	"net/http"
	"github.com/Project/entity"
	"github.com/gin-gonic/gin"
)

// POST /LabResult
func CreateLabResult(c *gin.Context) {
	var MedicalTech entity.MedicalTech
	var MedicalRecord entity.MedicalRecord
	var LabType entity.LabType
	var LabRoom entity.LabRoom
	var LabResult entity.LabResult

	// ผลลัพธ์ที่ได้จากขั้นตอนที่ 8 จะถูก bind เข้าตัวแปร LabResult
	if err := c.ShouldBindJSON(&LabResult); err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

	// 9: ค้นหา MedicalTech ด้วย id
	if tx := entity.DB().Where("id = ?",
LabResult.MedicalTechID).First(&MedicalTech); tx.RowsAffected == 0 {
		c.JSON(http.StatusBadRequest, gin.H{"error": "MedicalTech
not found"})
		return
	}

	// 10: ค้นหา MedicalRecord ด้วย id
	if tx := entity.DB().Where("id = ?",
LabResult.MedicalRecordID).First(&MedicalRecord); tx.RowsAffected == 0 {
		c.JSON(http.StatusBadRequest, gin.H{"error": "MedicalRecord
not found"})
		return
	}

	// 11: ค้นหา LabType ด้วย id
	if tx := entity.DB().Where("id = ?",
LabResult.LabTypeID).First(&LabType); tx.RowsAffected == 0 {
		c.JSON(http.StatusBadRequest, gin.H{"error": "LabType not
found"})
		return
	}
	//12: ค้นหา LabRoom ด้วย id
```

```
	if tx := entity.DB().Where("id = ?",
LabResult.LabRoomID).First(&LabRoom); tx.RowsAffected == 0 {
		c.JSON(http.StatusBadRequest, gin.H{"error": "LabRoom not
found"})
		return
	}
	// 13: สร้าง LabResult
	lr := entity.LabResult{
		MedicalTech:  MedicalTech,	//Mt
		MedicalRecord:	MedicalRecord, //Mr
		LabType:	LabType,	//Type
		Lab_Result: LabResult.Lab_Result,
		Lab_Detail: LabResult.Lab_Detail,
		LabRoom: LabRoom,	//Room
		AddedTime: LabResult.AddedTime,
	}

	// 14: บันทึก
	if err := entity.DB().Create(&lr).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	c.JSON(http.StatusOK, gin.H{"data": lr})
}

// GET: /api/ListLabResult
func ListLabResult(c *gin.Context) {
	var LabResult []*entity.LabResult
	if err :=
entity.DB().Preload("MedicalTech").Preload("MedicalRecord").Preload("Lab
Type").Preload("LabRoom").Table("lab_results").Find(&LabResult).Error;
err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

	c.JSON(http.StatusOK, gin.H{"data": LabResult})
}
```

Source ของ GORM

ไฟล์ entity/MedicalRecord.go

```
package entity

import (
	"time"

	"gorm.io/gorm"
)

//MedicalRecord
type MedicalRecordOfficer struct {
	gorm.Model
	MedRecOfficer_Name  string
	MedRecOfficer_Email string `gorm:"uniqueIndex"`
	MedRecOfficer_Pass  string
	MedicalRecord       []MedicalRecord
`gorm:"foreignKey:MedRecOfficerID"`
}

type NameTitle struct {
	gorm.Model
	Title         string
	MedicalRecord []MedicalRecord `gorm:"foreignKey:NameTitleID"`
}

type HealthInsurance struct {
	gorm.Model
	HealthInsurance_Name string
	Detail               string
	MedicalRecord        []MedicalRecord
`gorm:"foreignKey:HealthInsuranceID"`
}

type MedicalRecord struct {
	gorm.Model

	Hospital_Number string `gorm:"uniqueIndex"`
	Personal_ID     string `gorm:"uniqueIndex"`

	NameTitleID *uint
	NameTitle   NameTitle

	Patient_Name  string
	Patient_Age   int
	Patient_dob   time.Time
	Patient_Tel   string
	Register_Date time.Time
```

```
      HealthInsuranceID *uint
      HealthInsurance   HealthInsurance

      MedRecOfficerID *uint
      MedRecOfficer   MedicalRecordOfficer

      // เชื่อมกับ Screenings
      Screenings []Screening
`gorm:"foreignKey:MedRecID;constraint:OnDelete:CASCADE"`

      // เชื่อมกับ DrugAllergies
      DrugAllergies []DrugAllergy
`gorm:"foreignKey:MedicalRecordID;constraint:OnDelete:CASCADE"`

      //เชื่อมกับ MedicalHistory
      MedicalHistories []MedicalHistory
`gorm:"foreignKey:MedicalRecordID;constraint:OnDelete:CASCADE"`

      //เชื่อมกับ refers
      Refers []Refer
`gorm:"foreignKey:MedicalRecordID;constraint:OnDelete:CASCADE"`

      //เชื่อมกับ LabResult
      LabResults []LabResult
`gorm:"foreignKey:MedicalRecordID;constraint:OnDelete:CASCADE"`
}
```

ไฟล์ entity/LabResult.go

```
package entity
import (
	"time"
	"gorm.io/gorm"
)
//LabResult
type MedicalTech struct {
	gorm.Model
	Name       string
	Email      string `gorm:"uniqueIndex"`
	Pass       string
	LabResults []LabResult `gorm:"foreignKey:MedicalTechID"`
}
type LabType struct {
	gorm.Model
	Name       string
	LabResults []LabResult `gorm:"foreignKey:LabTypeID"`
}
type LabRoom struct {
	gorm.Model
	Name       string
	Building   string
	floor      int
	LabResults []LabResult `gorm:"foreignKey:LabRoomID"`
}

type LabResult struct {
	gorm.Model

	MedicalTechID *uint
	MedicalTech   MedicalTech

	MedicalRecordID *uint
	MedicalRecord   MedicalRecord

	LabTypeID *uint
	LabType   LabType

	Lab_Result string
	Lab_Detail string

	LabRoomID *uint
	LabRoom   LabRoom
	AddedTime time.Time
}
```

ไฟล์ main.go

```
package main

import (
	Actor "github.com/Project/controller/Actor"
	Disease "github.com/Project/controller/Disease"
	DrugAllergy "github.com/Project/controller/DrugAllergy"
	LabResult "github.com/Project/controller/LabResult"
	MedicalHistory "github.com/Project/controller/MedicalHistory"
	MedicalRecord "github.com/Project/controller/MedicalRecord"
	Refer "github.com/Project/controller/Refer"
	Screening "github.com/Project/controller/Screening"
	"github.com/Project/entity"
	"github.com/Project/middlewares"
	"github.com/gin-gonic/gin"
)

func main() {
	entity.SetupDatabase()

	r := gin.Default()
	r.Use(CORSMiddleware())
	api := r.Group("")
	{
		protected := api.Use(middlewares.Authorizes())
		{
			//api Diseases
			protected.GET("/api/ListDiseases",
Disease.ListDiseases)

			//api DrugAllergy
			protected.POST("/api/CreateDrugAllergy",
DrugAllergy.CreateDrugAllergy)
			protected.GET("/api/ListDrugAllergy",
DrugAllergy.ListDrugAllergy)
			protected.GET("/api/ListDrug", DrugAllergy.ListDrug)

			//api MedicalRecord
			protected.GET("/api/ListMedicalRecord",
MedicalRecord.ListMedicalRecord)
			protected.GET("/api/ListHealthInsurance",
MedicalRecord.ListHealthInsurance)
			protected.GET("/api/ListNameTitle",
MedicalRecord.ListNameTitle)
			protected.POST("/api/CreateMedicalRecord",
MedicalRecord.CreateMedicalRecord)

			//api MedicalHistory
```

```
                  protected.GET("/api/ListDepartments",
MedicalHistory.ListDepartments)
                  protected.POST("/api/CreateMedicalHistory",
MedicalHistory.CreateMedicalHistory)
                  protected.GET("/api/ListMedicalHistories",
MedicalHistory.ListMedicalHistories)

                  //api Refer
                  protected.GET("/api/ListHospitals",
Refer.ListHospitals)
                  protected.POST("/api/CreateRefer", Refer.CreateRefer)
                  protected.GET("/api/ListRefer", Refer.ListRefer)

                  //api Screening
                  protected.POST("/api/CreateScreening",
Screening.CreateScreening)
                  protected.GET("/api/ListScreenings",
Screening.ListScreenings)

                  //api ListLabResult
                  protected.GET("/api/ListLabType",
LabResult.ListLabType)
                  protected.GET("/api/ListLabRoom",
LabResult.ListLabRoom)
                  protected.POST("/api/CreateLabResult",
LabResult.CreateLabResult)
                  protected.GET("/api/ListLabResult",
LabResult.ListLabResult)

            }
      }
      //Get func login/Actor
      r.POST("/api/LoginDoctor", Actor.LoginDoctor)
      r.POST("/api/LoginMedicalRecordOfficer",
Actor.LoginMedicalRecordOfficer)
      r.POST("/api/LoginMedicalTech", Actor.LoginMedicalTech)
      r.POST("/api/LoginNurse", Actor.LoginNurse)

      // Run the server
      r.Run()
}
func CORSMiddleware() gin.HandlerFunc {
      return func(c *gin.Context) {
            c.Writer.Header().Set("Access-Control-Allow-Origin", "*")
            c.Writer.Header().Set("Access-Control-Allow-Credentials",
"true")
            c.Writer.Header().Set("Access-Control-Allow-Headers",
"Content-Type, Content-Length, Accept-Encoding, X-CSRF-Token,
Authorization, accept, origin, Cache-Control, X-Requested-With")
```

```
            c.Writer.Header().Set("Access-Control-Allow-Methods", "POST,
OPTIONS, GET, PUT")

            if c.Request.Method == "OPTIONS" {
                  c.AbortWithStatus(204)
                  return
            }

            c.Next()
      }
}
```

ไฟล์ middlewares/ authorizes.go

```
package middlewares

import (
	"net/http"
	"strings"
	"github.com/Project/service"
	"github.com/gin-gonic/gin"
)

// validates token
func Authorizes() gin.HandlerFunc {
	return func(c *gin.Context) {
		clientToken := c.Request.Header.Get("Authorization")
		if clientToken == "" {
			c.JSON(http.StatusForbidden, gin.H{"error": "No
Authorization header provided"})
			return
		}

		extractedToken := strings.Split(clientToken, "Bearer ")

		if len(extractedToken) == 2 {
			clientToken = strings.TrimSpace(extractedToken[1])
		} else {
			c.JSON(http.StatusBadRequest, gin.H{"error":
"Incorrect Format of Authorization Token"})
			return
		}

		jwtWrapper := service.JwtWrapper{
			SecretKey: "SvNQpBN8y3qlVrsGAYYWoJJk56LtzFHx",
			Issuer:    "AuthService",
		}

		claims, err := jwtWrapper.ValidateToken(clientToken)
		if err != nil {
			c.JSON(http.StatusUnauthorized, gin.H{"error":
err.Error()})
			return
		}

		c.Set("email", claims.Email)
		c.Next()

	}
}
```

ไฟล์ Service/ authentication.go

```
package service

import (
	"errors"
	"time"

	jwt "github.com/dgrijalva/jwt-go"
)

// JwtWrapper wraps the signing key and the issuer
type JwtWrapper struct {
	SecretKey       string
	Issuer          string
	ExpirationHours int64
}

// JwtClaim adds email as a claim to the token
type JwtClaim struct {
	Email string
	jwt.StandardClaims
}

// Generate Token generates a jwt token
func (j *JwtWrapper) GenerateToken(email string) (signedToken string,
err error) {
	claims := &JwtClaim{
		Email: email,
		StandardClaims: jwt.StandardClaims{
			ExpiresAt: time.Now().Local().Add(time.Hour *
time.Duration(j.ExpirationHours)).Unix(),
			Issuer:    j.Issuer,
		},
	}

	token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)

	signedToken, err = token.SignedString([]byte(j.SecretKey))
	if err != nil {
		return
	}

	return
}

//Validate Token validates the jwt token
func (j *JwtWrapper) ValidateToken(signedToken string) (claims
*JwtClaim, err error) {
	token, err := jwt.ParseWithClaims(
```

```
            signedToken,
            &JwtClaim{},
            func(token *jwt.Token) (interface{}, error) {
                  return []byte(j.SecretKey), nil
            },
      )

      if err != nil {
            return
      }

      claims, ok := token.Claims.(*JwtClaim)
      if !ok {
            err = errors.New("Couldn't parse claims")
            return
      }

      if claims.ExpiresAt < time.Now().Local().Unix() {
            err = errors.New("JWT is expired")
            return
      }

      return

}
```

ไฟล์ entity/Setup.go

```
package entity

import (
	"time"

	"golang.org/x/crypto/bcrypt"
	"gorm.io/driver/sqlite"
	"gorm.io/gorm"
)

var db *gorm.DB

func DB() *gorm.DB {
	return db
}

func SetupDatabase() {
	database, err := gorm.Open(sqlite.Open("sa-64-project.db"),
&gorm.Config{})
	if err != nil {
		panic("failed to connect database")
	}

	database.AutoMigrate(
		&MedicalRecord{}, &MedicalRecordOfficer{}, &NameTitle{},
&HealthInsurance{},
		&Nurse{}, &Disease{}, &Screening{},
		&Drug{}, &DrugAllergy{},
		&Doctor{}, &Department{}, &MedicalHistory{},
		&Hospital{}, &Refer{},
		&MedicalTech{}, &LabType{}, &LabRoom{}, &LabResult{},
	)

	db = database

	//setup MedicalRecord
	PasswordMedicalRecordOfficer1, err :=
bcrypt.GenerateFromPassword([]byte("111111a"), 14)
	MedicalRecordOfficer1 := MedicalRecordOfficer{
		MedRecOfficer_Name:  "Rosie",
		MedRecOfficer_Email: "rosie@gmail.com",
		MedRecOfficer_Pass:  string(PasswordMedicalRecordOfficer1),
	}
	db.Model(&MedicalRecordOfficer{}).Create(&MedicalRecordOfficer1)

	PasswordMedicalRecordOfficer2, err :=
bcrypt.GenerateFromPassword([]byte("2222222a"), 14)
	MedicalRecordOfficer2 := MedicalRecordOfficer{
```

```
            MedRecOfficer_Name:  "Carla",
            MedRecOfficer_Email: "carla@gmail.com",
            MedRecOfficer_Pass:  string(PasswordMedicalRecordOfficer2),
      }
      db.Model(&MedicalRecordOfficer{}).Create(&MedicalRecordOfficer2)

      // setup nametitle
      NameTitle1 := NameTitle{
            Title: "นาง",
      }
      db.Model(&NameTitle{}).Create(&NameTitle1)

      NameTitle2 := NameTitle{
            Title: "นางสาว",
      }
      db.Model(&NameTitle{}).Create(&NameTitle2)

      NameTitle3 := NameTitle{
            Title: "นาย",
      }
      db.Model(&NameTitle{}).Create(&NameTitle3)

      NameTitle4 := NameTitle{
            Title: "เด็กชาย",
      }
      db.Model(&NameTitle{}).Create(&NameTitle4)

      NameTitle5 := NameTitle{
            Title: "เด็กหญิง",
      }
      db.Model(&NameTitle{}).Create(&NameTitle5)


      //setup HealthInsurance
      HealthInsurance1 := HealthInsurance{
            HealthInsurance_Name: "นักศึกษา",
            Detail:               "นักศึกษามหาวิทยาลัยเทคโนโลยีสุรนารีรักษาฟรี",
      }
      db.Model(&HealthInsurance{}).Create(&HealthInsurance1)

      HealthInsurance2 := HealthInsurance{
            HealthInsurance_Name: "บัตรทอง",
            Detail:               "สวัสดิการแห่งรัฐ 30 บาทรักษาทุกโรค",
      }
      db.Model(&HealthInsurance{}).Create(&HealthInsurance2)

      //setup MedicalRecord1
      MedicalRecord1 := MedicalRecord{
            Hospital_Number: "2001",
```

```
            Personal_ID:     "1234567891234",
            Patient_Name:    "Saifon",
            Patient_Age:     21,
            Patient_dob:     time.Now(),
            Patient_Tel:     "0823642199",
            Register_Date:   time.Now(),
            HealthInsurance: HealthInsurance2,
            MedRecOfficer:   MedicalRecordOfficer2,
            NameTitle:       NameTitle2,
      }
      db.Model(MedicalRecord{}).Create(&MedicalRecord1)

      MedicalRecord2 := MedicalRecord{
            Hospital_Number: "2002",
            Personal_ID:     "9876543210123",
            Patient_Name:    "Sainam",
            Patient_Age:     26,
            Patient_dob:     time.Now(),
            Patient_Tel:     "0987475566",
            Register_Date:   time.Now(),
            HealthInsurance: HealthInsurance1,
            MedRecOfficer:   MedicalRecordOfficer1,
            NameTitle:       NameTitle3,
      }
      db.Model(MedicalRecord{}).Create(&MedicalRecord2)

      //setup Disease
      Disease1 := Disease{
            Name:        "Dengue",
            Description: "illnesses that cause fever, aches and pains,
or a rash. The most common symptom of dengue is fever",
      }
      db.Model(&Disease{}).Create(&Disease1)

      Disease2 := Disease{
            Name:        "Heart attack",
            Description: "the chest can feel like it's being pressed or
squeezed by a heavy object, and pain can radiate from the chest to the
jaw, neck, arms and back",
      }
      db.Model(&Disease{}).Create(&Disease2)

      Disease3 := Disease{
            Name:        "Gastritis",
            Description: "Gnawing or burning ache or pain (indigestion)
in your upper abdomen that may become either worse or better with
eating",
      }
      db.Model(&Disease{}).Create(&Disease3)
```

```
	//setup Nurse
	PasswordNurse1, err := bcrypt.GenerateFromPassword([]byte("1234"),
14)
	Nurse1 := Nurse{
		Name:  "pakapon seepakdee",
		Email: "pakapon@gmail.com",
		Pass:  string(PasswordNurse1),
	}
	db.Model(&Nurse{}).Create(&Nurse1)

	PasswordNurse2, err := bcrypt.GenerateFromPassword([]byte("123"),
14)
	Nurse2 := Nurse{
		Name:  "kritsada papakdee",
		Email: "big16635@gmail.com",
		Pass:  string(PasswordNurse2),
	}
	db.Model(&Nurse{}).Create(&Nurse2)

	PasswordNurse3, err :=
bcrypt.GenerateFromPassword([]byte("123456789"), 14)
	Nurse3 := Nurse{
		Name:  "somying kondee",
		Email: "somying@gmail.com",
		Pass:  string(PasswordNurse3),
	}
	db.Model(&Nurse{}).Create(&Nurse3)


	//setup Drug
	Drug1 := Drug{
		Drug_Name:       "Amantadine",
		Drug_properties: "รักษาไข้หวัดใหญ่สายพันธุ์เอ และโรคพาร์กินสัน",
		Drug_group:      "ยาต้านไวรัส",
		Stock:           20,
	}
	db.Model(&Drug{}).Create(&Drug1)

	Drug2 := Drug{
		Drug_Name:       "Lithium",
		Drug_properties: "บรรเทาหรือป้องกันการเกิดซ้ำของอาการจากโรคอารมณ์สองขั้ว ภาวะอารมณ์ดีตื่นตัว
ผิดปกติ",
		Drug_group:      "ยารักษาโรคจิต (Antipsychotics)",
		Stock:           500,
	}
	db.Model(&Drug{}).Create(&Drug2)

	//setup MedicalTech
	PassMedicalTech1, err :=
bcrypt.GenerateFromPassword([]byte("123"), 14)
```

```
	PassMedicalTech2, err :=
bcrypt.GenerateFromPassword([]byte("12345"), 14)

	MedicalTech1 := MedicalTech{
		Name:  "Chalermkiet kongkapan",
		Email: "chalermkiet@gmail.com",
		Pass:  string(PassMedicalTech1),
	}
	db.Model(&MedicalTech{}).Create(&MedicalTech1)

	MedicalTech2 := MedicalTech{
		Name:  "Somkiat Kongkapan",
		Email: "Somkiat@gmail.com",
		Pass:  string(PassMedicalTech2),
	}
	db.Model(&MedicalTech{}).Create(&MedicalTech2)

	//setup LabType
	LabType1 := LabType{
		Name: "Blood test",
	}
	db.Model(&LabType{}).Create(&LabType1)

	LabType2 := LabType{
		Name: "Urinalysis test",
	}
	db.Model(&LabType{}).Create(&LabType2)

	//setup LabRoom
	LabRoom1 := LabRoom{
		Name:     "Hematology",
		Building: "Rattanavejjapat",
		floor:    1,
	}
	db.Model(&LabRoom{}).Create(&LabRoom1)

	LabRoom2 := LabRoom{
		Name:     "Microbiology",
		Building: "Research Center",
		floor:    7,
	}
	db.Model(&LabRoom{}).Create(&LabRoom2)

	//setup Doctor
	PasswordDoctor1, err :=
bcrypt.GenerateFromPassword([]byte("yhyh555"), 14)
	Doctor1 := Doctor{
		Name:     "Yohan Song",
		Tel:      "0885556699",
		Email:    "yh.s@hp.ac.th",
```

```
            Password: string(PasswordDoctor1),
      }
      db.Model(&Doctor{}).Create(&Doctor1)

      PasswordDoctor2, err :=
bcrypt.GenerateFromPassword([]byte("password1234"), 14)
      Doctor2 := Doctor{
            Name:     "Supot Jamsai",
            Tel:      "0877774412",
            Email:    "supot.j@hp.ac.th",
            Password: string(PasswordDoctor2),
      }
      db.Model(&Doctor{}).Create(&Doctor2)

      PasswordDoctor3, err :=
bcrypt.GenerateFromPassword([]byte("123456789"), 14)
      Doctor3 := Doctor{
            Name:     "Suchawadee Teangtrong",
            Tel:      "0644416289",
            Email:    "Suchawadee@gmail.com",
            Password: string(PasswordDoctor3),
      }
      db.Model(&Doctor{}).Create(&Doctor3)

      //setup Department
      Department1 := Department{
            Name:     "Emergency Room",
            Building: "Surapiphat",
            Floor:    1,
      }
      db.Model(&Department{}).Create(&Department1)

      Department2 := Department{
            Name:     "Radiology Department",
            Building: "Thepnipa",
            Floor:    5,
      }
      db.Model(&Department{}).Create(&Department2)

      Department3 := Department{
            Name:     "Pediatrics Department",
            Building: "Pataranavee",
            Floor:    5,
      }
      db.Model(&Department{}).Create(&Department3)

      //setup Hospital
      Hospital1 := Hospital{
            Name: "Srithanya Hospital",
            Tel:  "025287800",
```

```
	}
	db.Model(&Hospital{}).Create(&Hospital1)

	Hospital2 := Hospital{
		Name: "Suranaree Hospital",
		Tel:  "020000000",
	}
	db.Model(&Hospital{}).Create(&Hospital2)



}
```