G-12 B6210571 นายอภิสิทธิ์ สะโมรส

ระบบหลัก: ระบบทันตกรรม **ระบบย่อย:** ระบบบันทึกข้อมูลการรักษาทางทันตกรรม

1. Requirements

ระบบทันตกรรมของโรงพยาบาลเป็นระบบที่ให้ผู้ใช้งานระบบซึ่งเป็นบุคลกรทางการแพทย์สามารถเข้าใช้งานระบบได้ โดยระบบทันตกรรมของโรงพยาบาลสามารถบันทึกข้อมูลเวชระเบียนได้ ในขั้นตอนการรับบริการผู้รับการรักษาจะทำการคัดกรองแล้วจะได้ใบคัดกรองข้อมูลพื้นฐาน ซึ่งจะทำการแยกประเภทการรักษาทำให้เกิดความสะดวกรวดเร็วขึ้น และในการรักษานั้นจะมีการให้ใบจ่ายยาและเวชภัณฑ์ในการตรวจรักษา โดยข้อมูลในการรักษานั้นจะถูกทำการบันทึกไว้ในฐานข้อมูลโดยทันตแพทย์ เพื่อเก็บไว้ติดตามการรักษาเมื่อจำเป็น

User Story (ระบบบันทึกข้อมูลการรักษาทางทันตกรรม *)

ในบทบาทของ	ทันตแพทย์
ฉันต้องการ	ให้ระบบสามารถบันทึกข้อมูลการวินิจฉัย ในครั้งนั้น ๆ เข้าไปไว้ในฐานข้อมูล
เพื่อ	ให้ฉันสามารถเพิ่มหรือค้นหาข้อมูลใบวินิจฉัย ของผู้ที่มารับการรักษาได้

Output หน้าจอ ผู้ใช้งานระบบกดเข้าใบวินิจฉัย => ระบบแสดงฟอร์มสำหรับการบันทึกข้อมูลการวินิจฉัย เมื่อทำการกรอกข้อมูลเรียบร้อยแล้ว ระบบจะการแสดงข้อความขึ้นเพื่อบอกว่าทำการบันทึกสำเร็จหรือไม่

Output ข้อมูล ระบบบันทึกข้อมูลใบวินิจฉัยเก็บไว้ในฐานข้อมูล

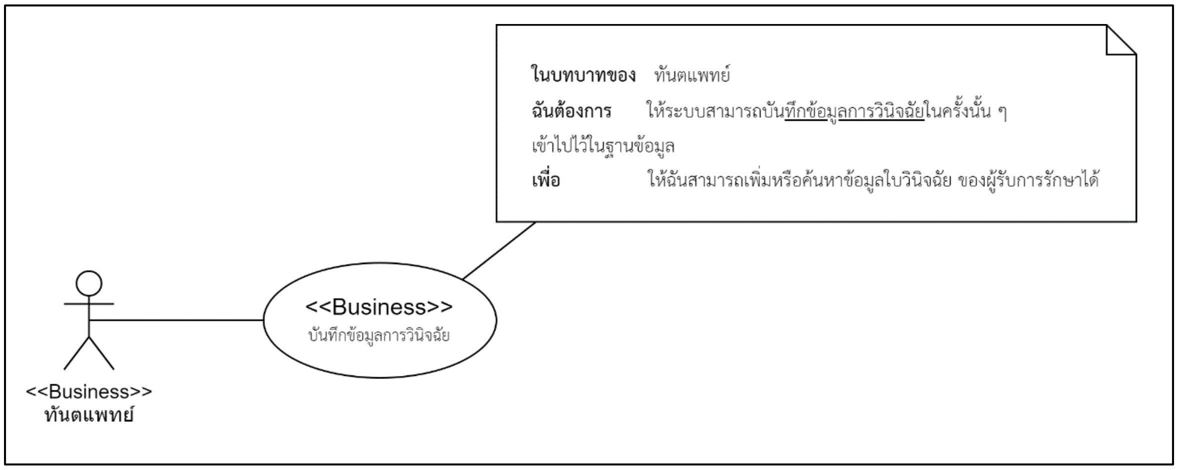
คำนามที่อาจจะกลายเป็น Entity (ตารางในฐานข้อมูล) ระบบบันทึกข้อมูลการรักษาทางทันตกรรม

คำนาม	**เหตุผล: เกี่ยวข้องกับ User Story หรือไม่**
เวชระเบียน	เกี่ยวข้อง แต่ไม่โดยตรง
ใบคัดกรองข้อมูลพื้นฐาน	เกี่ยวข้องโดยตรง เพราะต้องใช้เป็นตัวอ้างอิงในการรักษา
ทันตแพทย์	เกี่ยวข้องโดยตรง เพราะว่าเป็นตัวข้อมูลบทบาท
ประเภทการรักษา	เกี่ยวข้องโดยตรง เพราะว่าต้องทราบว่าทำการรักษาอะไรไป
ใบวินิจฉัย	เกี่ยวข้องโดยตรง เพราะเป็นสิ่งที่ต้องบันทึกลงฐานข้อมูล

G-12 B6210571 นายอภิสิทธิ์ สะโมรส

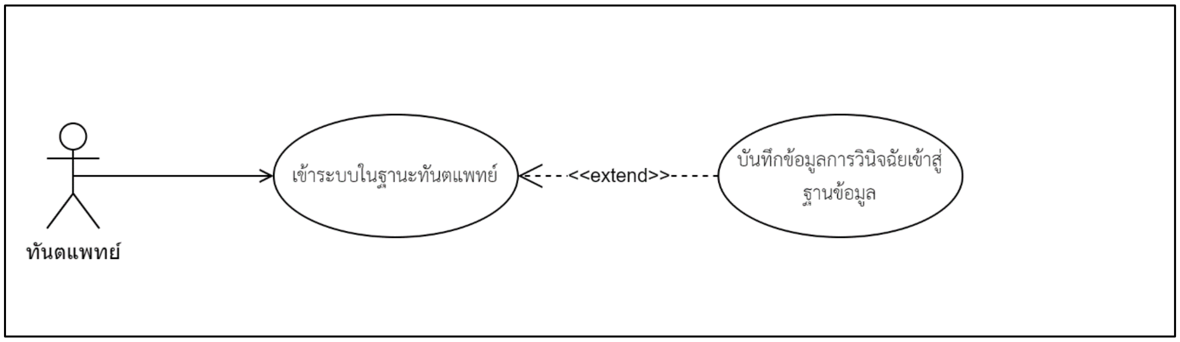
ระบบหลัก: ระบบทันตกรรม **ระบบย่อย:** ระบบบันทึกข้อมูลการรักษาทางทันตกรรม

2. Business Use Case

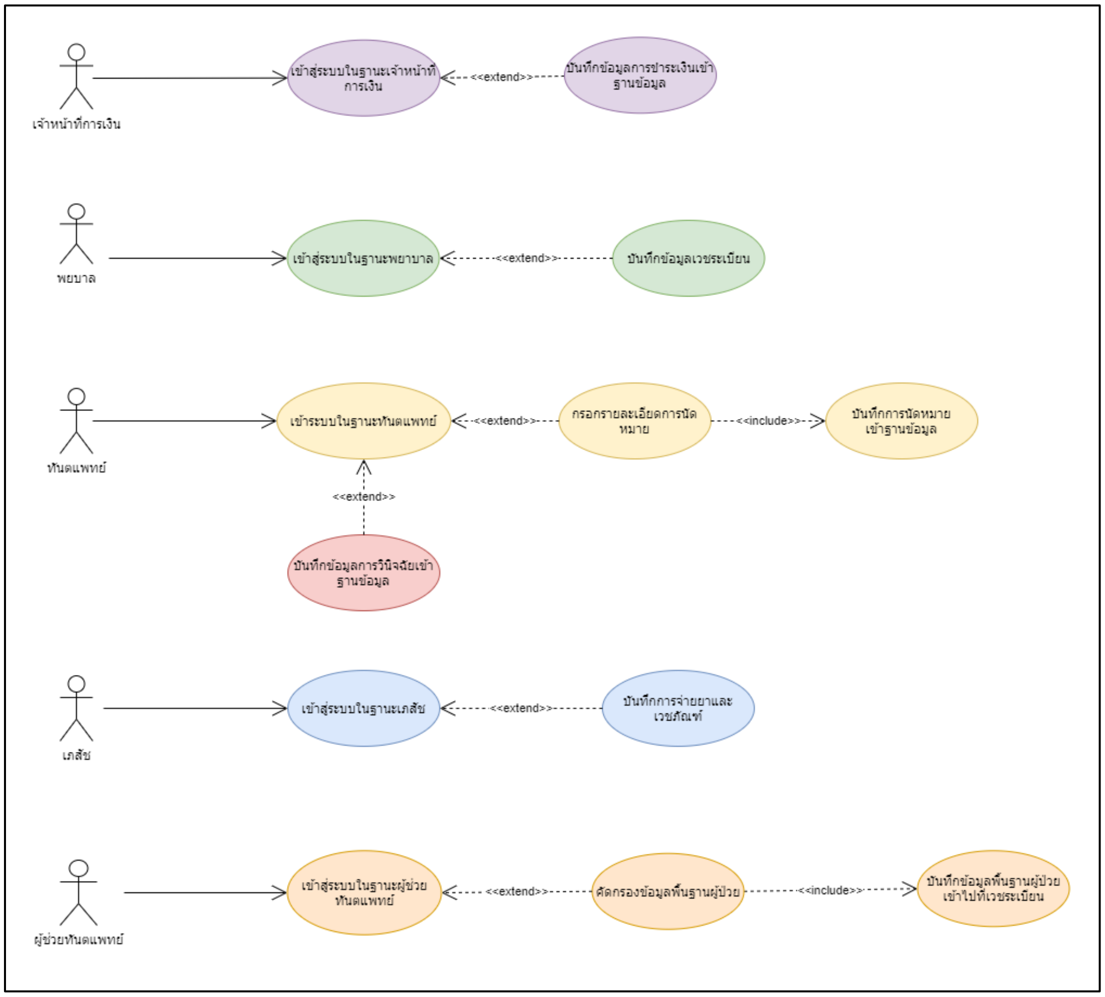


ในบทบาทของ ทันตแพทย์
ฉันต้องการ ให้ระบบสามารถบันทึกข้อมูลการวินิจฉัยในครั้งนั้น ๆ เข้าไปไว้ในฐานข้อมูล
เพื่อ ให้ฉันสามารถเพิ่มหรือค้นหาข้อมูลใบวินิจฉัย ของผู้รับการรักษาได้
<<Business>>
บันทึกข้อมูลการวินิจฉัย
<<Business>>
ทันตแพทย์

3. System Use Case



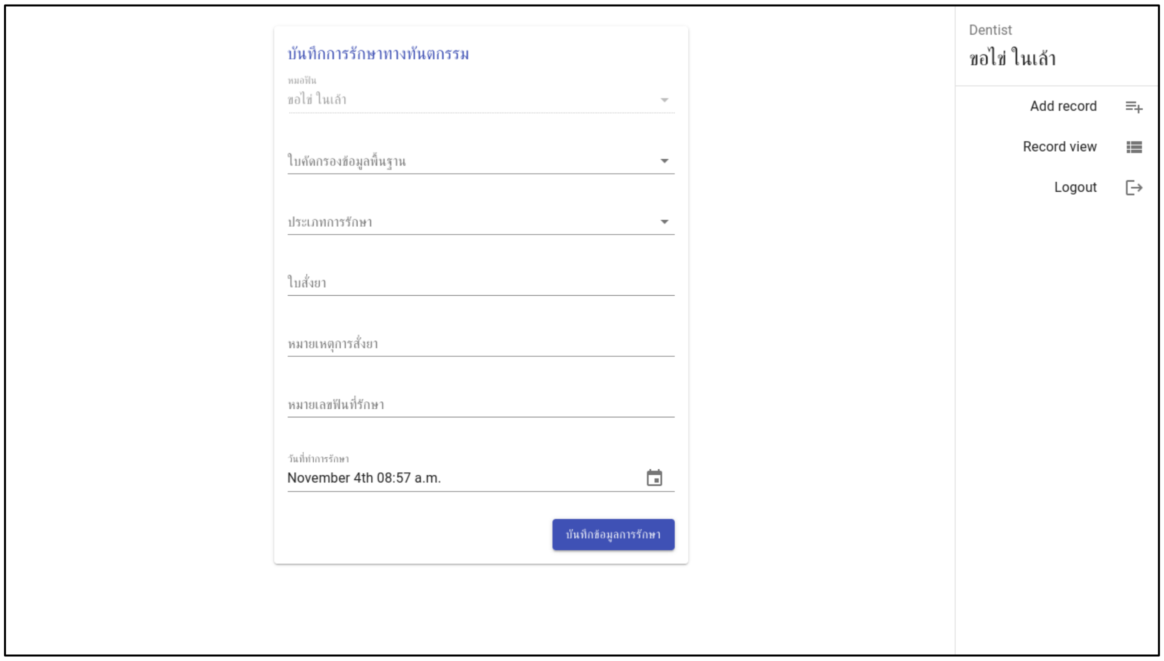
4. System Use Case รวมทั้งระบบใหญ่ในหน้าเดียว


เจ้าหน้าที่การเงิน
เข้าสู่ระบบในฐานะเจ้าหน้าที่การเงิน
<<extend>>
บันทึกข้อมูลการชำระเงินเข้าฐานข้อมูล
พยาบาล
เข้าสู่ระบบในฐานะพยาบาล
<<extend>>
บันทึกข้อมูลเวชระเบียน
ทันตแพทย์
เข้าระบบในฐานะทันตแพทย์
<<extend>>
กรอกรายละเอียดการนัดหมาย
<<include>>
บันทึกการนัดหมายเข้าฐานข้อมูล
<<extend>>
บันทึกข้อมูลการวินิจฉัยเข้าฐานข้อมูล
เภสัช
เข้าสู่ระบบในฐานะเภสัช
<<extend>>
บันทึกการจ่ายยาและเวชภัณฑ์
ผู้ช่วยทันตแพทย์
เข้าสู่ระบบในฐานะผู้ช่วยทันตแพทย์
<<extend>>
คัดกรองข้อมูลพื้นฐานผู้ป่วย
<<include>>
บันทึกข้อมูลพื้นฐานผู้ป่วยเข้าไปที่เวชระเบียน

G-12 B6210571 นายอภิสิทธิ์ สะโมรส

ระบบหลัก: ระบบทันตกรรม **ระบบย่อย:** ระบบบันทึกข้อมูลการรักษาทางทันตกรรม

<u>5. User Interface</u>



บันทึกการรักษาทางทันตกรรม

หมอฟัน: ขอไข่ ในเล้า

ใบคัดกรองข้อมูลพื้นฐาน

ประเภทการรักษา

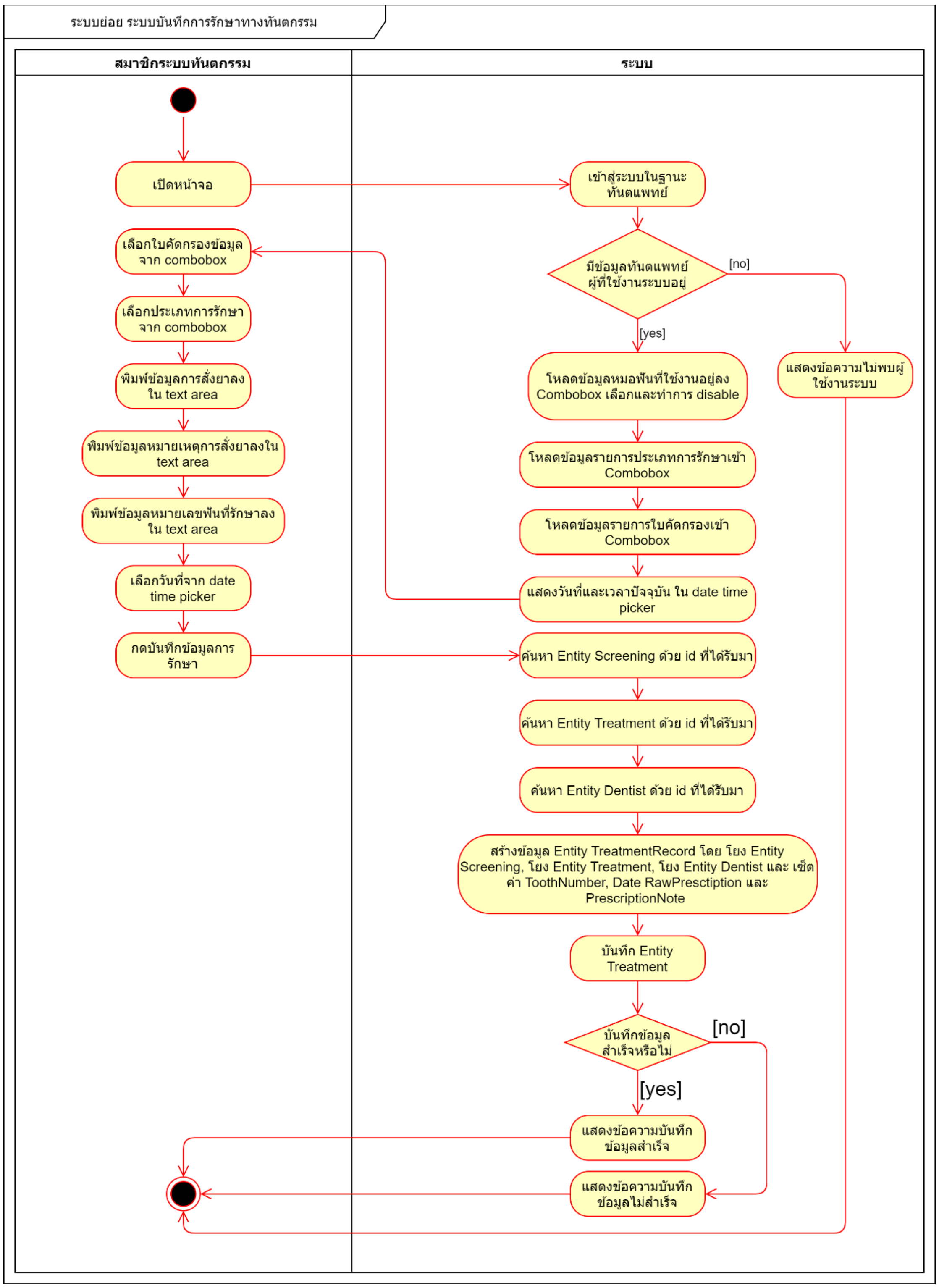
ใบสั่งยา

หมายเหตุการสั่งยา

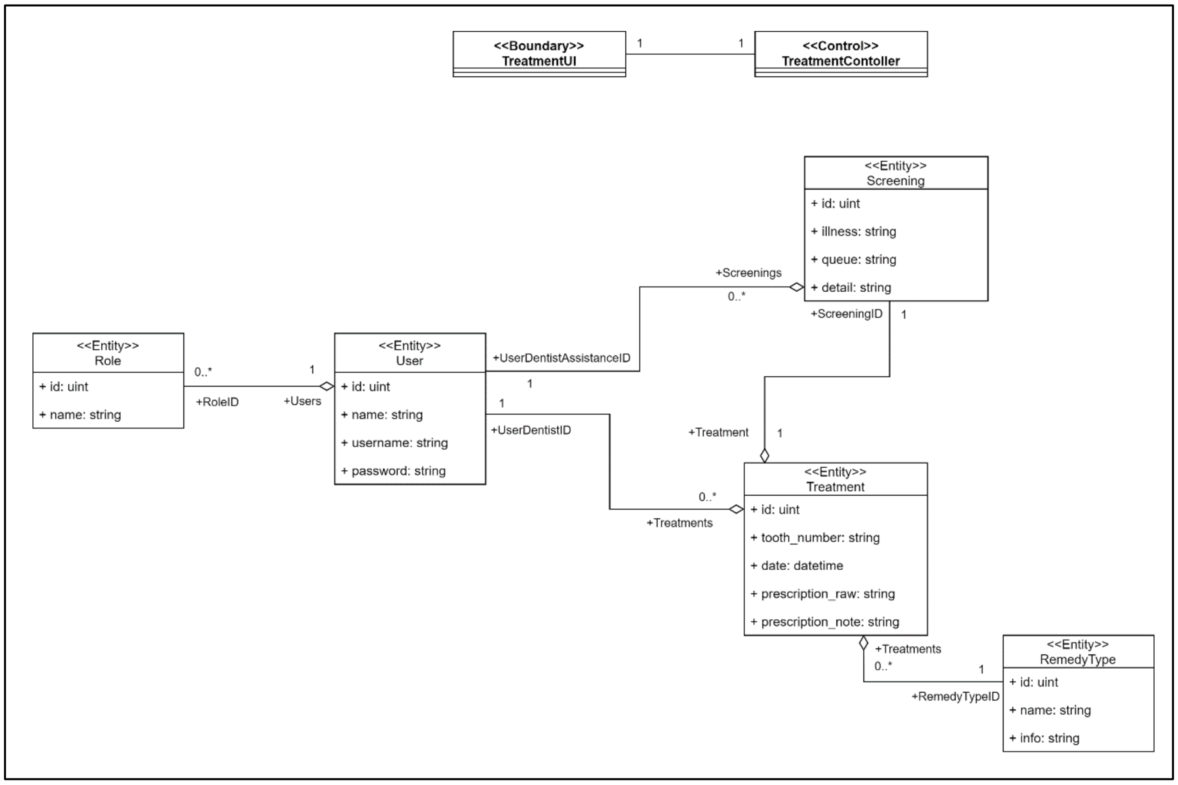
หมายเลขฟันที่รักษา

วันที่ทำการรักษา: November 4th 08:57 a.m.

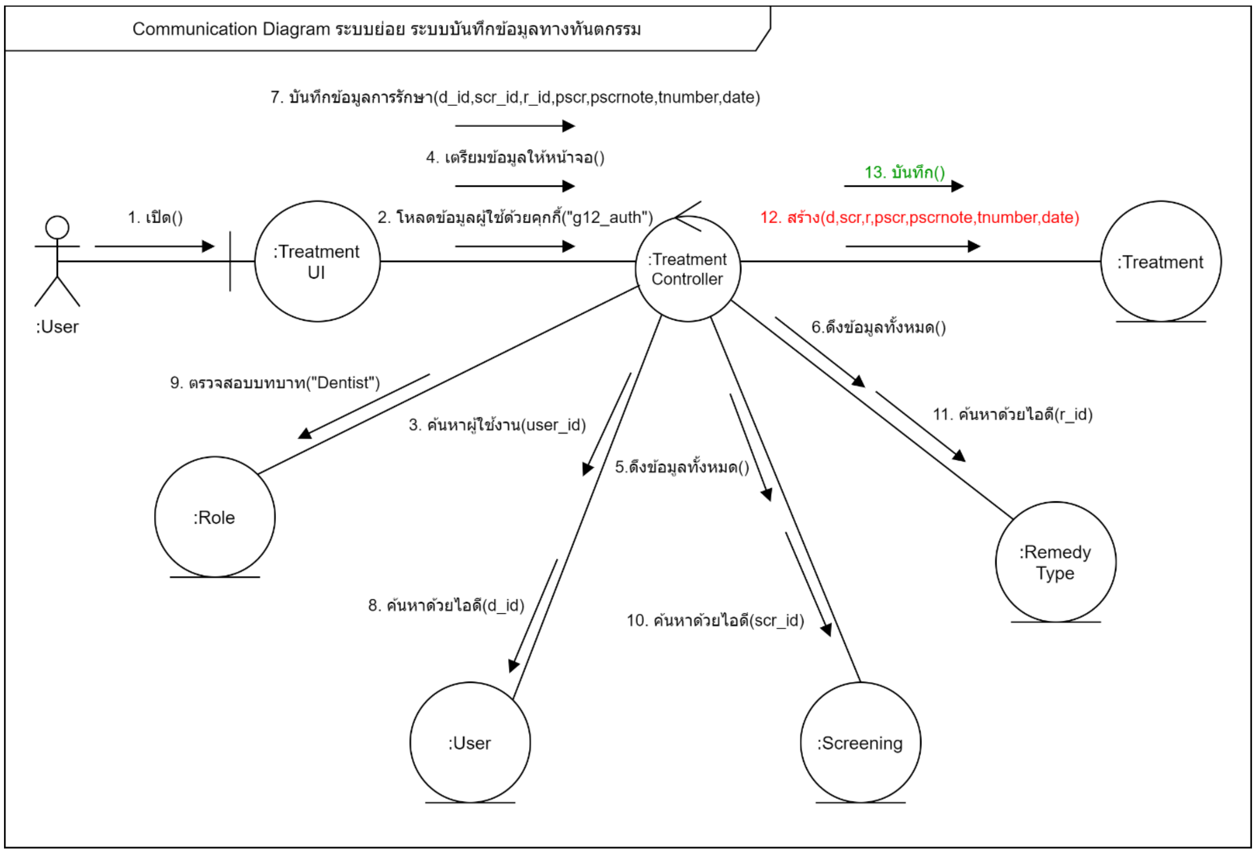
6. System Activity Diagram


ระบบย่อย ระบบบันทึกการรักษาทางทันตกรรม
สมาชิกระบบทันตกรรม
ระบบ
เปิดหน้าจอ
เข้าสู่ระบบในฐานะทันตแพทย์
มีข้อมูลทันตแพทย์ผู้ที่ใช้งานระบบอยู่
[no]
[yes]
แสดงข้อความไม่พบผู้ใช้งานระบบ
โหลดข้อมูลหมอฟันที่ใช้งานอยู่ลง Combobox เลือกและทำการ disable
โหลดข้อมูลรายการประเภทการรักษาเข้า Combobox
โหลดข้อมูลรายการใบคัดกรองเข้า Combobox
แสดงวันที่และเวลาปัจจุบัน ใน date time picker
เลือกใบคัดกรองข้อมูล จาก combobox
เลือกประเภทการรักษา จาก combobox
พิมพ์ข้อมูลการสั่งยาลง ใน text area
พิมพ์ข้อมูลหมายเหตุการสั่งยาลงใน text area
พิมพ์ข้อมูลหมายเลขฟันที่รักษาลง ใน text area
เลือกวันที่จาก date time picker
กดบันทึกข้อมูลการรักษา
ค้นหา Entity Screening ด้วย id ที่ได้รับมา
ค้นหา Entity Treatment ด้วย id ที่ได้รับมา
ค้นหา Entity Dentist ด้วย id ที่ได้รับมา
สร้างข้อมูล Entity TreatmentRecord โดย โยง Entity Screening, โยง Entity Treatment, โยง Entity Dentist และ เซ็ตค่า ToothNumber, Date RawPresctiption และ PrescriptionNote
บันทึก Entity Treatment
บันทึกข้อมูลสำเร็จหรือไม่
[no]
[yes]
แสดงข้อความบันทึกข้อมูลสำเร็จ
แสดงข้อความบันทึกข้อมูลไม่สำเร็จ

7. Class Diagram ระดับ Design



8. Communication Diagram (เฉพาะ Action หลัก)


Communication Diagram ระบบย่อย ระบบบันทึกข้อมูลทางทันตกรรม
7. บันทึกข้อมูลการรักษา(d_id,scr_id,r_id,pscr,pscrnote,tnumber,date)
4. เตรียมข้อมูลให้หน้าจอ()
13. บันทึก()
1. เปิด()
2. โหลดข้อมูลผู้ใช้ด้วยคุกกี้("g12_auth")
12. สร้าง(d,scr,r,pscr,pscrnote,tnumber,date)
:User
:Treatment UI
:Treatment Controller
:Treatment
6.ดึงข้อมูลทั้งหมด()
9. ตรวจสอบบทบาท("Dentist")
3. ค้นหาผู้ใช้งาน(user_id)
11. ค้นหาด้วยไอดี(r_id)
5.ดึงข้อมูลทั้งหมด()
:Role
:Remedy Type
8. ค้นหาด้วยไอดี(d_id)
10. ค้นหาด้วยไอดี(scr_id)
:User
:Screening

9. Source Code

React path /frontend/

./src/App.tsx

```
import React from 'react';
import { BrowserRouter as Router, Switch, Route } from 'react-router-dom';
import LoginPage from './components/Loginpage';
import TRMcomponent from './components/TRMcomponent';

function App() {

      return (
      <Router>
            <link
                  rel="stylesheet"
      href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&
display=swap"
            />
            <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons" />
            <div>
                  <Switch>
                        <Route exact path="/" component={LoginPage} />
                        <Route exact component={TRMcomponent} />
                  </Switch>
            </div>
      </Router>
  );
}

export default App;
```

./src/models/IPatient.ts

```
export default interface PatientInterface {
      ID                : string;
      Firstname   : string;
      Lastname    : string;
}
```

. /src/models/IRemedyType.ts

```
export default interface RemedyTypeInterface  {
      ID          : string,
      Name  : string
}
```

. /src/models/IScreening.ts

```
import PatientInterface from "./IPatient"
export default interface ScreeningInterface {
      ID          : string;
      Queue       : string;
      Patient           : PatientInterface;
}
```

./src/models/index.d.ts

```
import { Color as AlertColor } from '@material-ui/lab/Alert'
import ScreeningInterface from "./IScreening"
import PatientInterface from "./IPatient"
import RemedyTypeInterface from "./IRemedyType"

interface UserLogin {
      ID : string;
      RoleID : string;
      Name : string;
      RoleName : string;
}

interface AlertInfo {
      message : string;
      level : AlertColor;
}

export type {
      ScreeningInterface,
      UserInterface,
      RemedyTypeInterface,
      UserLogin,
      AlertInfo,
      PatientInterface,
}

export interface TreatmentInteface {
      ID : string;
      Date: Date;
      ToothNumber: string;
      Screening : ScreeningInterface;
      RemedyType: RemedyTypeInterface;
}
```

./src/components/Utils.tsx

```
import React from 'react'
import {
      Typography, Icon,
      Drawer, List, ListItem, Divider, ListItemText, ListItemIcon
} from '@material-ui/core';
import { createStyles, makeStyles, Theme} from '@material-ui/core/styles';
import { UserLogin } from "../models";
import * as H from 'history'
import { useHistory } from 'react-router';

export const Auth = async (history : H.History, isLoginPage: boolean,
setUser: React.Dispatch<React.SetStateAction<UserLogin>> ) => {
      const apiUrl = 'http://localhost:8080/TRMauth';
      const requestOption = {
            method : "GET",
            header : { "Content-Type" : "application/json" },
            credentials : 'include' as RequestCredentials,
      };

      fetch(apiUrl, requestOption)
      .then((response) => response.json())
      .then((res) => {
            if (res.data) {
                  if ( isLoginPage )
                        history.replace("/treatment_record_data", res.data)
                  setUser(res.data)
            } else {
                  history.replace("/")
            }
      });
}

const drawerWidth = '240px';

const useStyles = makeStyles( (them:Theme) => createStyles({
      card: {
            display: 'flex',
            flexDirection: 'column',
            width: '12vw',
            minHeight: '10vh',
            position: 'fixed',
            top: 0,
            right: 0,
            margin: '20px'
      },
      drawer:
      {
            width: drawerWidth,
            flexShrink: 0,
      },
      drawerPaper: {
            width: drawerWidth,
      },
}))

export const UserCard = (props: {data:UserLogin}) => {
      const classes = useStyles();
      const history = useHistory();

      function Logout() {
            const apiUrl = 'http://localhost:8080/TRMlogout';
            const requestOption = {
                  method : "GET",
                  header : { "Content-Type" : "application/json" },
                  credentials : 'include' as RequestCredentials,
            };

            fetch(apiUrl, requestOption)
            .then((response) => response.json())
            .then((res) => {
                  if (res.data) {
                        history.replace("")
                  }
            });
      }

      return (
            <React.Fragment>

            <Drawer
                  className={classes.drawer}
                  variant="permanent"
                  classes={{
                        paper: classes.drawerPaper,
                  }}
                  anchor="right"
            >
                  <List>
                  <ListItem style={{flexDirection: 'column', alignItems:
'start', paddingBottom:'16px'}}>
                        <Typography color="textSecondary" gutterBottom>
                              {props.data.RoleName}
                        </Typography>
                        <Typography variant="h5" component="h2">
                              {props.data.Name}
                        </Typography>
                  </ListItem>
                  <Divider/>
                  <ListItem button onClick={ () =>
history.replace('/treatment_record') }>
                        <ListItemText style={{
display:'flex',justifyContent:'flex-end'}} >Add record</ListItemText>
                        <ListItemIcon
                              style={{ justifyContent:'flex-end'}}
                        ><Icon>playlist_add</Icon></ListItemIcon>
                  </ListItem>
                  <ListItem button onClick={ () =>
history.replace('/treatment_record_data') }>
                        <ListItemText style={{
display:'flex',justifyContent:'flex-end'}} >Record view</ListItemText>
                        <ListItemIcon
                              style={{ justifyContent:'flex-end'}}
                        ><Icon>view_list</Icon></ListItemIcon>
                  </ListItem>
                  <ListItem button onClick={ Logout }>
                        <ListItemText style={{
display:'flex',justifyContent:'flex-end'}} >Logout</ListItemText>
                        <ListItemIcon
                              style={{ justifyContent:'flex-end'}}
                        ><Icon>logout</Icon></ListItemIcon>
                  </ListItem>
                  </List>
            </Drawer>

            </React.Fragment>
      )
}
```

./src/components/TRMcomponent.tsx

```
import React, { useEffect } from 'react';
import { Route } from 'react-router-dom';

import TreatmentRecord from './TreatmentRecord'
import TreatmentData from './TreatmentData';
import { Auth, UserCard } from './Utils';
import { useHistory } from "react-router";
import {UserLogin} from '../models'

function TRMcomponent() {

      const history = useHistory();
      const [user, setUser] = React.useState<UserLogin>({
            ID: "", Name: "", RoleID: "", RoleName: ""
      });

      useEffect(()=>{
            Auth( history, false, setUser )
      },[])

      return (
            <div>
                  <UserCard data={user} />

                  <Route exact path="/treatment_record" render={ ()=><
TreatmentRecord user={user}/> }  />
                  <Route exact path="/treatment_record_data"
component={TreatmentData} />
            </div>
  );
}

export default TRMcomponent;
```

./src/components/Loginpage.tsx

```
import React, {useEffect} from 'react'
import { useHistory } from 'react-router-dom';
import {
      Typography, Button, FormControl, Container,
      Box, Divider, Snackbar, Card,
      InputLabel, Input, InputAdornment, IconButton, Icon
} from '@material-ui/core';
import MuiAlert, { AlertProps } from '@material-ui/lab/Alert'
import {
      Visibility, VisibilityOff, AccountCircle
} from '@material-ui/icons'

import { createStyles, makeStyles, Theme } from '@material-ui/core/styles';
import { Auth } from './Utils'
import { UserLogin, AlertInfo } from '../models';

// css style classes
const useStyles = makeStyles( (them:Theme) => createStyles({
      root: {
            display: 'flex',
            flexDirection: 'column',
            width: '100vw',
            height: '100vh',
            justifyContent: 'center',
            alignItems: 'center',
      },
      textForm: {
            margin: '8px',
            display: 'flex',
            maxWidth: '25ch',
      },
      loginButton: {
            display: 'flex',
            justifyContent: 'flex-end',
            padding: '16px 0px 0px 0px',
            margin: '8px',
            maxWidth: '25ch',
      },
      card : {
            padding: '0px 8px',
      },
      head : {
            margin: '8px 8px -4px 8px',
      }
}))

// interface for user login data
interface UserData {
      Username: string;
      Pass: string;
      showPassword: boolean;
}

// interface for alert props


// custom alert I supposed
function Alert(props: AlertProps) {
      return <MuiAlert elevation={6} variant="filled" {...props} />;
}

// this is login page
// no props required
export default function LoginPage() {
      const classes = useStyles();
      const history = useHistory();

      //----------------------------//
      // Value user passed in
      const [values, setValues] = React.useState<UserData>({
            Username: '',
            Pass: '',
            showPassword: false,
      });
      const handleChange = (prop: keyof UserData) => (event:
React.ChangeEvent<HTMLInputElement>) => {
            setValues({ ...values, [prop]: event.target.value });
      };

      const handleClickShowPassword = () => {
            setValues({ ...values, showPassword: !values.showPassword });
      };

      const handleMouseDownPassword = (event:
React.MouseEvent<HTMLButtonElement>) => {
            event.preventDefault();
      };
      //----------------------------//
      ////////////////////////////////
      //----------------------------//
      // Alert for user notice
      const [status, setStatus] = React.useState(false);
      const [message, setMessage] =
React.useState<AlertInfo>({message:'',level: 'warning'});

      const Login = async () => {
            let data = {
                  Username : values.Username,
                  Password : values.Pass
            }
            const apiUrl = 'http://localhost:8080/TRMlogin';
            const requestOption = {
                  method : "POST",
                  header : { "Content-Type" : "application/json" },
                  credentials : 'include' as RequestCredentials,
                  body : JSON.stringify(data)
            };

            fetch(apiUrl, requestOption)
            .then((response) => response.json())
            .then((res) => {
                  if (res.data) {
                        history.replace("/treatment_record_data", {})
                  } else {
                        setStatus(true);
                        setMessage({message:"Incorrect Username or 
Password", level:'error'});
                  }
            });
      };

      const handleClick = () => {
            // no input in field
            if ( values.Username === "" || values.Pass === "" ) {
                  setStatus(true);
                  setMessage({message:"Empty username or password", 
level:'warning'});
            } else {
                  Login();
            }
      };

      const handleClose = (event: React.SyntheticEvent | React.MouseEvent, 
reason?: string) => {
            if (reason === 'clickaway') return;
            setStatus(false);
      };
      //----------------------------//

      const [user, setUser] = React.useState<UserLogin>({
            ID: "", Name: "", RoleID: "", RoleName: ""
      });

      useEffect(()=> {
            Auth(history, true, setUser);
      }, []);
      return (
            <React.Fragment>
                  <Snackbar open={status} autoHideDuration={6000}
onClose={handleClose}>
                  <Alert severity={message.level} onClick={handleClose}>
{message.message} </Alert>
                  </Snackbar>
                  <Box className={classes.root} >
                        <Card className={classes.card}>
                        <Typography className={classes.head}
variant="h6">G12 Dental-Clinic</Typography>
                        <Divider/>
                        <FormControl className={classes.textForm}>
                              <InputLabel>Username</InputLabel>
                              <Input
                                    type='text'
                                    value={values.Username}
                                    onChange={handleChange('Username')}
                                    endAdornment={
                                          <InputAdornment position="end">
                                          <IconButton disabled>
<AccountCircle/> </IconButton>
                                          </InputAdornment>
                                    }/>
                        </FormControl>
                        <FormControl className={classes.textForm}>
                              <InputLabel>Password</InputLabel>
                              <Input
                                    type={values.showPassword ? 'text' :
'password'}
                                    value={values.Pass}
                                    onChange={handleChange('Pass')}
                                    endAdornment={
                                          <InputAdornment position="end">
                                          <IconButton
                                                aria-label="toggle password
visibility"

      onClick={handleClickShowPassword}

      onMouseDown={handleMouseDownPassword}
                                          >
                                          {values.showPassword ?
<Visibility /> : <VisibilityOff />}
                                          </IconButton>
                                          </InputAdornment>
                                    }/>
                        </FormControl>
                        <Container className={classes.loginButton}>
                              <Button
                                    variant="contained"
                                    color="primary"

      endIcon={<Icon>arrow_forward_ios</Icon>}
                                    size="small"
                                    onClick={handleClick}
                              > Login </Button>
                              </Container>
                        </Card>
                  </Box>
            </React.Fragment>
      );
}
```

./src/components/TreatmentData.tsx

```
import React, {useEffect} from 'react'
import { TreatmentInteface } from '../models';
import { createStyles, makeStyles, Theme  } from '@material-
ui/core/styles';
import {
      TableCell, TableBody, TableContainer, TableHead, TableRow, Paper,
Table, Container
} from '@material-ui/core';

const useStyles = makeStyles((theme:Theme) =>
      createStyles({
            root: {
                  width: 'calc(100% - 240px)',
                  marginRight: '240px',
                  padding: '24px'
            },
            table: {},
      })
);

export default function TreatmentData( ) {
      const classes = useStyles();
      // treatment-data list handler
      const [treatments, setTreatments] =
React.useState<TreatmentInteface[]>([]);
      const getTreatments = async () => {
            const apiUrl = "http://localhost:8080/treatmentRecords";
            const requestOption = {
                  method : "GET",
                  header : { "Content-Type" : "application/json" }
            };
            fetch(apiUrl, requestOption)
            .then((response) => response.json())
            .then((res) => {
                  console.log(res.data);
                  if (res.data) setTreatments(res.data);
            });
      }
      useEffect(()=>{
            getTreatments();
      }, []);

      return (
            <React.Fragment>
                  <Container className={classes.root}>
                  <TableContainer component={Paper} >
                  <Table className={classes.table} aria-label="simple
table">
                        <TableHead>
                        <TableRow>
                              <TableCell>RecordID</TableCell>
                              <TableCell align="left">Name</TableCell>
                              <TableCell
align="right">Treatment</TableCell>
                              <TableCell align="right">Date</TableCell>
                              <TableCell align="right">Tooth
Number</TableCell>
                        </TableRow>
                        </TableHead>
                        <TableBody>
                        {treatments.map((treatment) => (
                              <TableRow key={treatment.ID}>
                              <TableCell component="th"
scope="row">{treatment.ID}</TableCell>
                              <TableCell
align="left">{treatment.Screening.Patient.Firstname}
{treatment.Screening.Patient.Lastname}</TableCell>
                              <TableCell
align="right">{treatment.RemedyType.Name}</TableCell>
                              <TableCell align="right">{new Date(
treatment.Date ).toLocaleString("th-TH")}</TableCell>
                              <TableCell
align="right">{treatment.ToothNumber}</TableCell>
                              </TableRow>
                        ))}
                        </TableBody>
                  </Table>
                  </TableContainer>
                  </Container>
            </React.Fragment>
      );
}
```

./src/components/TreatmentRecord.tsx

```
import React, { useEffect } from 'react';
import { createStyles, makeStyles, Theme } from '@material-ui/core/styles';
import {
	Typography, Button, TextField, FormControl, Container,
	Paper, Grid, Box, Snackbar, Select, MenuItem,
	InputLabel,
} from '@material-ui/core';
import MuiAlert, { AlertProps } from "@material-ui/lab/Alert";
import { MuiPickersUtilsProvider, KeyboardDateTimePicker } from "@material-
ui/pickers";
import DateFnsUtils from "@date-io/date-fns";
import {
	ScreeningInterface, RemedyTypeInterface, UserLogin, AlertInfo
} from "../models";

function Alert(props: AlertProps) {
	return <MuiAlert elevation={6} variant="filled" {...props} />;
}

const useStyles = makeStyles((theme:Theme) =>
	createStyles({
		root: {
			width: 'calc(100% - 240px)',
			margin: '0px 240px 0px 0px',
			padding: '24px',
			flexGrow: 1
		},
		container: {
			width: '50%',
		},
		paper: {
			padding: theme.spacing(2),
			color: theme.palette.text.secondary
		},
		formControl: {
			margin: theme.spacing(0),
			minWidth: "100%",
		},
		selectEmpty: {
			marginTop: theme.spacing(2),
		},
	})
);

interface treatmentFields {
	rawPrescription: string;
	prescriptionInfo: string;
	toothNumber: string;
```

```
}


function TreatmentRecord( props: {user:UserLogin} ) {
      const classes = useStyles();

      const [otherData,setOtherData] = React.useState<treatmentFields>({
            rawPrescription: "",
            prescriptionInfo: "",
            toothNumber: "",
      })

      const handleDataChange = (prop: keyof treatmentFields ) => (event :
React.ChangeEvent<HTMLInputElement|HTMLTextAreaElement>) => {
            setOtherData( { ...otherData, [prop]:event.target.value } )
      }

      // date-time handler
      const [selectedDate, setSelectedDate] = React.useState<Date | null>
(new Date());
      const handleDateChange = (date:Date | null) => {
setSelectedDate(date); };

      // screenings list handler
      const [screenings, setScreenings ] =
React.useState<ScreeningInterface[]>([])
      const [selectedScreening, setScreening] = React.useState("");
      const handleScreeningChange = ( event:React.ChangeEvent<{ name?:
string; value: unknown }> ) =>{
            setScreening(event.target.value as string);
      };

      const getScreening = async () => {
            const apiUrl = "http://localhost:8080/screenings";
            const requestOption = {
                  method : "GET",
                  header : { "Content-Type" : "application/json" }
            };

            fetch(apiUrl, requestOption)
            .then((response) => response.json())
            .then((res) => {
                  console.log(res.data);
                  if (res.data) setScreenings(res.data);
            });
      }

      // remedy-types list handler
      const [remedyTypes, setRemedyTypes] = 
React.useState<RemedyTypeInterface[]>([]);
      const [selectedRemedy, setRemedy] = React.useState("");
      const handleRemedyTypeChange = ( event:React.ChangeEvent<{ name?: 
string; value: unknown }> ) =>{
            setRemedy(event.target.value as string);
      };

      const getRemedyTypes = async () => {
            const apiUrl = "http://localhost:8080/remedy_types";
            const requestOption = {
                  method : "GET",
                  header : { "Content-Type" : "application/json" }
            };

            fetch(apiUrl, requestOption)
            .then((response) => response.json())
            .then((res) => {
                  console.log(res.data);
                  if (res.data) setRemedyTypes(res.data);
            });
      }

      // popup handler
      const [open, setOpen] = React.useState(false);
      // use too simulate summit button
      const handleClose = (event?: React.SyntheticEvent, reason?: string) 
=> {
            if (reason === 'clickaway') {
                  return;
            }
            setOpen(false);
      };

      const [message, setMessage] = 
React.useState<AlertInfo>({message:'',level: 'warning'});

      function submit() {

            if ( props.user.RoleName !== "Dentist" ) {
                  setOpen(true);
                  setMessage( { message:'You have no authorize to make this 
action', level:'error'} );
                  return;
            }

            let data = {
                  PrescriptionRaw         : otherData.rawPrescription,
                  PrescriptionNote  : otherData.prescriptionInfo,
                  ToothNumber             : otherData.toothNumber,
                  Date                    : selectedDate,
                  ScreeningID             : selectedScreening,
                  UserDentistID           : props.user.ID,
                  RemedyTypeID            : selectedRemedy
            };

            const apiUrl = "http://localhost:8080/treatmentRecord";
            const requestOption = {
                  method : "POST",
                  header : { "Content-Type" : "application/json" },
                  body : JSON.stringify(data)
            };

            fetch(apiUrl, requestOption)
            .then((response) => response.json())
            .then((res) => {
                  console.log(res.data);
                  setOpen(true);
                  if (res.data) setMessage( { message:'Succesfully saved', 
level:'success' } );
                  else setMessage( { message:'Failed to save', 
level:'error' } );
            });
      }
      // load nesssecary data from database
      useEffect(()=> {
            getScreening();
            getRemedyTypes();
      }, []);

      return (
            <Container className={classes.root}>
                  <Snackbar open={open} autoHideDuration={6000} 
onClose={handleClose}>
                  <Alert severity={message.level} onClick={handleClose}>
{message.message} </Alert>
                  </Snackbar>
                  <Container className={classes.container}>
                  <Paper className={classes.paper}>
                        <Box display="flex">
                              <Box flexGrow={1}>
                                    <Typography
                                          component="h2"
                                          variant="h6"
                                          color="primary"
                                          gutterBottom
                                    >
                                          บันทึกการรักษาทางทันตกรรม
                                    </Typography>
                              </Box>
                        </Box>
                        <Grid container spacing={3} >
                              <Grid item xs={12}>
                                    <FormControl
className={classes.formControl} disabled error={ props.user.RoleName !==
"Dentist"}>
                                          <InputLabel id="demo-simple-
select-readonly-label">หมอฟัน</InputLabel>
                                          <Select
                                                labelId="demo-simple-
select-readonly-label"

                                                id="demo-simple-select-
readonly"

                                                inputProps={{ readOnly:
true }}

                                                value={props.user.ID}
                                                >
                                                <MenuItem
value={props.user.ID}>{props.user.Name}</MenuItem>
                                          </Select>
                                    </FormControl>
                              </Grid>
                              <Grid item xs={12}>
                                    <FormControl
className={classes.formControl}>
                                          <InputLabel id="demo-simple-
select-label">ใบคัดกรองข้อมูลพื้นฐาน</InputLabel>
                                          <Select
                                                labelId="demo-simple-
select-label"

                                                id="demo-simple-select"


      onChange={handleScreeningChange}

                                                >
                                                <MenuItem
value=""><em>None</em></MenuItem>
                                                { screenings.map(
(screening:ScreeningInterface) => (
                                                      <MenuItem
value={screening.ID} >
                                                            {screening.ID}
&nbsp;

      {screening.Queue} &nbsp;

      {screening.Patient.Firstname} &nbsp;
      {screening.Patient.Lastname}
                                                      </MenuItem>
                                                ))}
                                          </Select>
                                    </FormControl>
                              </Grid>
                              <Grid item xs={12}>
                                    <FormControl
className={classes.formControl}>
                                          <InputLabel id="demo-simple-
select-label">ประเภทการรักษา</InputLabel>
                                                <Select
                                                labelId="demo-simple-
select-label"
                                                id="demo-simple-select"

      onChange={handleRemedyTypeChange}
                                                >
                                                <MenuItem
value=""><em>None</em></MenuItem>
                                                { remedyTypes.map(
(remedyType:RemedyTypeInterface) => (
                                                      <MenuItem
value={remedyType.ID} >{remedyType.Name}</MenuItem>
                                                ))}
                                          </Select>
                                    </FormControl>
                              </Grid>
                              { [['ใบสั่งยา', 'rawPrescription'],['หมายเหตุการสั่ง
ยา','prescriptionInfo']].map( (prop: string[]) => (
                                    <Grid item xs={12}>
                                    <FormControl fullWidth
variant="outlined" >
                                          <TextField
                                                id="standard-multiline-
flexible"
                                                label={prop[0]}

      onChange={handleDataChange(prop[1] as keyof treatmentFields)}
                                                multiline
                                                maxRows={4}
                                                />
                                    </FormControl>
                                    </Grid>
                              )) }
                              <Grid item xs={12}>
                                    <FormControl fullWidth
variant="outlined" >
```

```
                                          <TextField id="standard-basic"
label="หมายเลขฟันที่รักษา" onChange={handleDataChange('toothNumber')}/>
                                    </FormControl>
                              </Grid>
                              <Grid item xs={12}>
                                    <FormControl fullWidth
variant="outlined" >
                                          <MuiPickersUtilsProvider
utils={DateFnsUtils}>
                                          <KeyboardDateTimePicker
                                                margin="normal"
                                                id="date-picker-dialog"
                                                label="วันที่ทำการรักษา"
                                                value={selectedDate}
                                                onChange={handleDateChange}
                                                KeyboardButtonProps={{
'aria-label': 'change date',}}
                                                />
                                          </MuiPickersUtilsProvider>
                                    </FormControl>
                              </Grid>
                              <Grid item xs={12}>
                                    <Button style={{float:"right"}}
variant="contained" color="primary" onClick={submit}>
                                          บันทึกข้อมูลการรักษา
                                    </Button>
                              </Grid>
                        </Grid>
                  </Paper>
                  </Container>
            </Container>
      );
}

export default TreatmentRecord;
```

Controller path /backend/

./services/TRMloginService.go

```
package service

import "golang.org/x/crypto/bcrypt"

type LoginService interface {
	Login(username string, password string)
}

type LoginData struct {
	Username string
	Password string
}

func (user *LoginData) Login(username string, password string) bool {
	return user.Username == username &&
(bcrypt.CompareHashAndPassword([]byte(password), []byte(user.Password)) ==
nil)
}
```

./controller/TRM_login.go

```
package controller

import (
	"fmt"
	"net/http"
	"strconv"
	"time"

	"github.com/ApisitSamorod/SA62G12/entity"
	service "github.com/ApisitSamorod/SA62G12/services"
	"github.com/dgrijalva/jwt-go"
	"github.com/gin-gonic/gin"
)

type UserData struct {
	RoleID   *uint
	ID       uint
	Name     string
	RoleName string
}

const trm_secretKey = "TMRJWT"

// POST /Login
// user login from login page
func TRM_LoginToSite(context *gin.Context) {
	var data service.LoginData
	var user entity.User
	if err := context.ShouldBindJSON(&data); err != nil {
		context.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
		return
	}

	if tx := entity.DB().Where("username = ?",
data.Username).First(&user); tx.RowsAffected == 0 {
		context.JSON(http.StatusBadRequest, gin.H{"error": "no match
user"})
		return
	}

	if !data.Login(user.Username, user.Password) { // so we find a user
		context.JSON(http.StatusBadRequest, gin.H{"error": "incorect
username or password"})
		return
	}

	claims := jwt.NewWithClaims(jwt.SigningMethodHS256,
jwt.StandardClaims{
		Issuer:    strconv.Itoa(int(user.ID)),
		ExpiresAt: time.Now().Add(time.Hour * 24).Unix(), //1 day
	})

	token, err := claims.SignedString([]byte(trm_secretKey))

	if err != nil {
		context.JSON(http.StatusInternalServerError, gin.H{"error":
"can't authorize credential"})
		return
	}

	http.SetCookie(context.Writer, &http.Cookie{
		Name:     "g12_auth",
		Value:    token,
		Expires:  time.Now().Add(time.Hour * 24),
		HttpOnly: true,
	})

	context.JSON(http.StatusOK, gin.H{"data": "successfully loggedin"})
}

// POST /auth
func TRM_CheckAuth(context *gin.Context) {
	var data UserData
	var user entity.User

	cookie, _ := context.Cookie("g12_auth")
	token, err := jwt.ParseWithClaims(cookie, &jwt.StandardClaims{},
func(token *jwt.Token) (interface{}, error) {
		return []byte(trm_secretKey), nil
	})

	if err != nil {
		context.JSON(http.StatusInternalServerError, gin.H{"error": "no
matched credential"})
		return
	}

	claims := token.Claims.(*jwt.StandardClaims)

	entity.DB().Joins("Role").Where("users.id =?",
claims.Issuer).First(&user)

	data = UserData{
		Name:     user.Name,
		ID:       user.ID,
		RoleID:   &user.Role.ID,
		RoleName: user.Role.Name,
	}

	context.JSON(http.StatusOK, gin.H{"data": data})
}

func TRM_Logout(context *gin.Context) {
	cookie, _ := context.Cookie("g12_auth")

	fmt.Println(cookie)

	http.SetCookie(context.Writer, &http.Cookie{
		Name:     "g12_auth",
		Expires:  time.Now().Add(-time.Hour),
		HttpOnly: true,
	})

	context.JSON(http.StatusOK, gin.H{"data": "logged out"})
}
```

./controller/remedyType.go

```
package controller

import (
	"net/http"

	"github.com/ApisitSamorod/SA62G12/entity"
	"github.com/gin-gonic/gin"
)
```

```
// POST /remedy_type
func CreateRemedyType(c *gin.Context) {
	var RemedyType entity.RemedyType

	if err := c.ShouldBindJSON(&RemedyType); err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

	if err := entity.DB().Create(&RemedyType).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

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

func GetRemedyType(c *gin.Context) {
	var RemedyType entity.RemedyType

	id := c.Param("id")
	if err := entity.DB().Raw("SELECT * FROM remedy_types WHERE id = ?", 
id).Scan(&RemedyType).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

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

// GET /remedy_types
func ListRemedyType(c *gin.Context) {
	var remedytypes []entity.RemedyType

	if err := entity.DB().Raw("SELECT * FROM 
remedy_types").Find(&remedytypes).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}

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

./controller/screening.go

```
package controller

import (
	"github.com/ApisitSamorod/SA62G12/entity"
	"github.com/gin-gonic/gin"

	"net/http"
)

// POST /screening
func CreateScreening(c *gin.Context) {

	var screening_record entity.Screening
	var patient entity.Patient
	var medical_product entity.MedicalProduct
	var dentistass entity.User

	//10:ผลลัพธ์ที่ได้จากขั้นตอนที่ x จะถูก bind เข้าตัวแปร scr
	if err := c.ShouldBindJSON(&screening_record); err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	//31:ค้นหา User ด้วย id
	if tx := entity.DB().Where("id = ?",
screening_record.UserDentistassID).First(&dentistass); tx.RowsAffected == 0
{
		c.JSON(http.StatusBadRequest, gin.H{"error": "Dentist not
found"})
		return
	}
	entity.DB().Joins("Role").Find(&dentistass)

	if dentistass.Role.Name != "Dentist" {
		c.JSON(http.StatusBadRequest, gin.H{"error": "only for
dentsit"})
		return
	}
	//7:ค้นหา patient ด้วย p_id
	if tx := entity.DB().Where("id = ?",
screening_record.PatientID).First(&patient); tx.RowsAffected == 0 {
		c.JSON(http.StatusBadRequest, gin.H{"error": "Patient not
found"})
		return
	}
	//11:ค้นหา medical_product ด้วย m_id
	if tx := entity.DB().Where("id = ?",
screening_record.MedicalProductID).First(&medical_product); tx.RowsAffected
== 0 {
		c.JSON(http.StatusBadRequest, gin.H{"error": "Medical Product
not found"})
		return
	}
	//12:สร้าง Screening_records(p_id, m_id, u_id, illnesses, detail)
	scr := entity.Screening{
		//โยงความสัมพันธ์กับ Entity Patient
		//โยงความสัมพันธ์กับ Entity Medical_product
		//โยงความสัมพันธ์กับ Entity User
		Patient:        patient,
		MedicalProduct: medical_product,
		UserDentistass: dentistass,
		Illnesses:      screening_record.Illnesses,
		Detail:         screening_record.Detail,
		Queue:          screening_record.Queue,
	}
	//13:บันทึก()
	if err := entity.DB().Create(&scr).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	c.JSON(http.StatusOK, gin.H{"data": scr})
}

//GET /screening
func GetScreening(c *gin.Context) {
	var screening_record entity.Screening
	id := c.Param("id")
	if err :=
entity.DB().Preload("Patient").Preload("MedicalProduct").Preload("UserDenti
stass").Raw("SELECT * FROM screenings WHERE id = ?",
id).Find(&screening_record).Error; err != nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	c.JSON(http.StatusOK, gin.H{"data": screening_record})
}

//GET /screenings
func ListScreening(c *gin.Context) {
	var screening_records []entity.Screening
	if err :=
entity.DB().Preload("Patient").Preload("MedicalProduct").Preload("UserDenti
stass").Raw("SELECT * FROM screenings").Find(&screening_records).Error; err
!= nil {
		c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
		return
	}
	c.JSON(http.StatusOK, gin.H{"data": screening_records})
}
```

./controller/treatment.go

```
package controller

import (
	"github.com/ApisitSamorod/SA62G12/entity"

	"github.com/gin-gonic/gin"

	"net/http"
)

// POST /treatmentRecord
func CreateTreatment(context *gin.Context) {
	var treatmentRecord entity.Treatment

	var screening entity.Screening
	var dentist entity.User
	var remedy entity.RemedyType

	if err := context.ShouldBindJSON(&treatmentRecord); err != nil {
		context.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
		return
	}

	if tx := entity.DB().Where("id = ?",
treatmentRecord.UserDentistID).First(&dentist); tx.RowsAffected == 0 {
		context.JSON(http.StatusBadRequest, gin.H{"error": "Dentist not
found"})
		return
	}

	entity.DB().Joins("Role").Find(&dentist)

	if dentist.Role.Name != "Dentist" {
		context.JSON(http.StatusBadRequest, gin.H{"error": "only for
dentsit"})
		return
	}

	if tx := entity.DB().Where("id = ?",
treatmentRecord.ScreeningID).First(&screening); tx.RowsAffected == 0 {
		context.JSON(http.StatusBadRequest, gin.H{"error": "Screening
not found"})
		return
	}

	if tx := entity.DB().Where("id = ?",
treatmentRecord.RemedyTypeID).First(&remedy); tx.RowsAffected == 0 {
		context.JSON(http.StatusBadRequest, gin.H{"error": "RemedyType
not found"})
		return
	}

	treatmentData := entity.Treatment{
		PrescriptionRaw:  treatmentRecord.PrescriptionRaw,
		PrescriptionNote: treatmentRecord.PrescriptionNote,
		ToothNumber:      treatmentRecord.ToothNumber,
		Date:             treatmentRecord.Date,
		// create with assosiation
		Screening:   screening,
		UserDentist: dentist,
		RemedyType:  remedy,
	}

	if err := entity.DB().Create(&treatmentData).Error; err != nil {
		context.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
		return
	}

	context.JSON(http.StatusOK, gin.H{"data": treatmentData})
}

// GET /treatmentRecords
func ListTreatmentRecord(context *gin.Context) {
	var treatmentRecords []entity.Treatment

	if err :=
entity.DB().Preload("Screening.Patient").Preload("RemedyType").Find(&treatm
entRecords).Error; err != nil {
		context.JSON(http.StatusBadRequest, gin.H{"error":
err.Error()})
		return
	}

	context.JSON(http.StatusOK, gin.H{"data": treatmentRecords})
}
```

GORM path /backend/

./entity/models.go

```
package entity

import (
	"time"

	"gorm.io/gorm"
)

type Role struct {
	gorm.Model
	Name  string
	Users []User `gorm:"foreignKey:RoleID"`
}

type User struct {
	gorm.Model
	Name     string
	Username string `gorm:"uniqueIndex"`
	Password string

	Patients   []Patient   `gorm:"foreignKey:UserNurseID"`
	Screenings []Screening `gorm:"foreignKey:UserDentistassID"`
	Treatments []Treatment `gorm:"foreignKey:UserDentistID"`
	Appoints   []Appoint   `gorm:"foreignKey:UserDentistID"`
	MedRecords []MedRecord `gorm:"foreignKey:UserPharmacistID"`
	Payments   []Payment   `gorm:"foreignKey:UserFinancialID"`

	RoleID *uint
	Role   Role
}

//ระบบย่อย ระบบบันทึกเวชระเบียน
type Job struct {
	gorm.Model
	Name string

	Patients []Patient `gorm:"foreignKey:JobID"`
}

type Insurance struct {
	gorm.Model
	Name   string
	Detail string

	Patients []Patient `gorm:"foreignKey:InsuranceID"`
}
```

```
type Sex struct {
	gorm.Model
	Name string

	Patients []Patient `gorm:"foreignKey:SexID"`
}

type Patient struct {
	gorm.Model
	Firstname string
	Lastname  string
	Age       int
	IDcard    string `gorm:"uniqueIndex"`
	Tel       string
	Time      time.Time

	UserNurseID *uint
	UserNurse   User

	JobID *uint
	Job   Job

	InsuranceID *uint
	Insurance   Insurance

	SexID *uint
	Sex   Sex

	Screenings []Screening `gorm:"foreignKey:PatientID"`
	Appoints   []Appoint   `gorm:"foreignKey:PatientID"`
	Payments   []Payment   `gorm:"foreignKey:PatientID"`
}

//ระบบย่อย ระบบคัดกรองข้อมูลพื้นฐานผู้ป่วย

type Screening struct {
	gorm.Model
	Illnesses string
	Detail    string
	Queue     string

	PatientID *uint
	Patient   Patient

	UserDentistassID *uint
	UserDentistass   User

	MedicalProductID *uint
	MedicalProduct   MedicalProduct
```

```
}

//ระบบย่อย ระบบบันทักการรักษาทางทันตกรรม
type Treatment struct {
      gorm.Model

      PrescriptionRaw  string
      PrescriptionNote string
      ToothNumber      string
      Date             time.Time

      ScreeningID *uint
      Screening   Screening

      UserDentistID *uint
      UserDentist   User

      RemedyTypeID *uint
      RemedyType   RemedyType

      MedRecords []MedRecord `gorm:"foreignKey:TreatmentID"`
}

//ระบบย่อย ระบบบันทึกการนัดหมาย
type RemedyType struct {
      gorm.Model
      Name      string
      Appoints  []Appoint   `gorm:"foreignKey:RemedyTypeID"`
      Treatment []Treatment `gorm:"foreignKey:RemedyTypeID"`
      Payments  []Payment   `gorm:"foreignKey:RemedyTypeID"`
}

type Appoint struct {
      gorm.Model
      AppointTime time.Time
      Todo        string

      UserDentistID *uint
      UserDentist   User

      PatientID *uint
      Patient   Patient

      RemedyTypeID *uint
      RemedyType   RemedyType
}

//ระบบย่อย ระบบบันทึกการจ่ายยาและเวชภัณฑ์
type MedicalProduct struct {
	gorm.Model
	Name       string
	Screenings []Screening `gorm:"foreignKey:MedicalProductID"`
	MedRecords []MedRecord `gorm:"foreignKey:MedicalProductID"`
}

type MedRecord struct {
	gorm.Model
	Amount uint

	TreatmentID *uint
	Treatment   Treatment

	UserPharmacistID *uint
	UserPharmacist   User

	MedicalProductID *uint
	MedicalProduct   MedicalProduct
}

//ระบบย่อย ระบบบันทึกการชำระเงิน
type Payment struct {
	gorm.Model

	Price   float32
	Paytime time.Time
	Note    string

	PatientID *uint
	Patient   Patient

	UserFinancialID *uint
	UserFinancial   User

	RemedyTypeID *uint
	RemedyType   RemedyType
}
```

./entity/setup.go

```
package entity

import (
	"time"

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

var db *gorm.DB
```

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

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

	// Migrate the schema
	database.AutoMigrate(
		&Role{}, &User{},
		&Job{}, &Insurance{}, &Patient{}, &Sex{},
		&Screening{},
		&Treatment{}, &RemedyType{},
		&Appoint{},
		&MedicalProduct{}, &MedRecord{},
		&Payment{},
	)

	db = database

	// ตำแหน่งงาน ------------------------------------------------------------------
	role1 := Role{
		Name: "Dentist",
	}
	db.Model(&Role{}).Create(&role1)

	role2 := Role{
		Name: "Dental assistant",
	}
	db.Model(&Role{}).Create(&role2)

	role3 := Role{
		Name: "Nurse",
	}
	db.Model(&Role{}).Create(&role3)

	role4 := Role{
		Name: "Pharmacist",
	}
	db.Model(&Role{}).Create(&role4)

	role5 := Role{
		Name: "Financial officer",
	}
	db.Model(&Role{}).Create(&role5)
	// รวมสมาชิกทุกตำแหน่ง >> entity User --------------------------------------
----------------
	password1, err := bcrypt.GenerateFromPassword([]byte("1234"), 14)
	password2, err := bcrypt.GenerateFromPassword([]byte("5678"), 14)
	dentist1 := User{
		Name:     "กอเอ๋ย กอไก่",
		Username: "nita",
		Password: string(password2),
		Role:     role1,
	}
	db.Model(&User{}).Create(&dentist1)

	dentist2 := User{
		Name:     "ขอไข่ ในเล้า",
		Username: "name",
		Password: string(password1),
		Role:     role1,
	}
	db.Model(&User{}).Create(&dentist2)

	dentistass1 := User{
		Name:     "คอควาย เข้านา",
		Username: "pitch",
		Password: string(password1),
		Role:     role2,
	}
	db.Model(&User{}).Create(&dentistass1)

	dentistass2 := User{
		Name:     "งองู ใจกล้า",
		Username: "kantapit",
		Password: string(password2),
		Role:     role2,
	}
	db.Model(&User{}).Create(&dentistass2)

	nurse1 := User{
		Name:     "จอจาน ใช้ดี",
		Username: "few",
		Password: string(password1),
		Role:     role3,
	}
	db.Model(&User{}).Create(&nurse1)

	nurse2 := User{
		Name:     "ฉอฉิ่ง ตีดัง",
		Username: "pcrc",
            Password: string(password2),
            Role:     role3,
      }
      db.Model(&User{}).Create(&nurse2)

      pharmacist1 := User{
            Name:     "ชอช้าง วิ่งหนี",
            Username: "fonthap",
            Password: string(password1),
            Role:     role4,
      }
      db.Model(&User{}).Create(&pharmacist1)

      pharmacist2 := User{
            Name:     "ซอโซ่ ล่ามดี",
            Username: "q1234",
            Password: string(password2),
            Role:     role4,
      }
      db.Model(&User{}).Create(&pharmacist2)

      financial1 := User{
            Name:     "ญอหญิง โสภา",
            Username: "tanodom",
            Password: string(password1),
            Role:     role5,
      }
      db.Model(&User{}).Create(&financial1)

      financial2 := User{
            Name:     "ฐอฐาน เข้ามารอง",
            Username: "s1234",
            Password: string(password2),
            Role:     role5,
      }
      db.Model(&User{}).Create(&financial2)

      // เพศ ---------------------------------------------------------------------------------------
      sex1 := Sex{
            Name: "ชาย",
      }
      db.Model(&Sex{}).Create(&sex1)

      sex2 := Sex{
            Name: "หญิง",
      }
      db.Model(&Sex{}).Create(&sex2)
```

```
	// อาชีพ -------------------------------------------------------------------------------------------
	job1 := Job{
		Name: "ราชการ",
	}
	db.Model(&Job{}).Create(&job1)

	job2 := Job{
		Name: "รัฐวิสหกิจ",
	}
	db.Model(&Job{}).Create(&job2)

	job3 := Job{
		Name: "นักศึกษา",
	}
	db.Model(&Job{}).Create(&job3)

	// สิทธิการรักษา -------------------------------------------------------------------------------------------
	insurance1 := Insurance{
		Name:   "สิทธิสวัสดิการข้าราชการ",
		Detail: "ข้าราชการและบคุคลในครอบครัวสามารถใช้สิทธิ์เบิกจ่ายตรง โดยใช้บัตรประชาชนในการเข้ารับบริการ
รักษาพยาบาลประเภทผู้ป่วยนอกทุกครั้ง ณ จุดชำระเงินโดยหากไม่ได้นำบัตรประชาชนมาแสดง หรือเอกสารที่กรมบัญชีกลางกำหนด ผู้รับบริการจะต้อง
สำรองจ่ายเงินค่ารักษาพยาบาลไปก่อน แล้วนำใบเสร็จรับเงินไปเบิกคืนกับส่วนราชการต้นสังกัด",
	}
	db.Model(&Insurance{}).Create(&insurance1)

	insurance2 := Insurance{
		Name:   "สิทธิประกันสังคม",
		Detail: "สามารถใช้สิทธิ์ได้เฉพาะกรณีที่มีใบส่งตัวมาจากโรงพยาบาลต้นสังกัด และชำระเงินสดเท่านั้น ยกเว้น กรณีมีใบ
ส่งตัวยืนยันการให้วางบิลโรงพยาบาลต้นสังกัดได้ ",
	}
	db.Model(&Insurance{}).Create(&insurance2)

	insurance3 := Insurance{
		Name:   "สิทธิหลักประกันสุขภาพ 30 บาท",
		Detail: "คุ้มครองบุคคลที่เป็นคนไทยมีเลขประจำตัวประชาชน 13 หลักที่ไม่ได้รับสิทธิสวัสดิการข้าราชการ หรือ สิทธิ
ประกันสังคม หรือสิทธิสวัสดิการรัฐวิสาหกิจ หรือสิทธิอื่น ๆ จากรัฐ",
	}
	db.Model(&Insurance{}).Create(&insurance3)

	// ประเภทการรักษา -------------------------------------------------------------------------------------------
	remedy1 := RemedyType{
		Name: "อุดฟัน",
	}
	db.Model(&RemedyType{}).Create(&remedy1)

	remedy2 := RemedyType{
		Name: "ขูดหินปูน",
	}
	db.Model(&RemedyType{}).Create(&remedy2)

	remedy3 := RemedyType{
		Name: "เอ็กซ์เรย์",
	}
	db.Model(&RemedyType{}).Create(&remedy3)

	// เวชระเบียน -----------------------------------------------------------------------------------------------
	patient1 := Patient{
		Firstname: "พัชรชาติ",
		Lastname:  "จิรศรีโสภา",
		Age:       20,
		IDcard:    "1329900000000",
		Tel:       "0902571569",
		Time:      time.Now(),
		Sex:       sex1,
		Job:       job3,
		Insurance: insurance3,
		UserNurse: nurse1,
	}
	db.Model(&Patient{}).Create(&patient1)

	patient2 := Patient{
		Firstname: "สมหญิง",
		Lastname:  "ซิ่งรถไถ",
		Age:       26,
		IDcard:    "1329900000001",
		Tel:       "0808571549",
		Time:      time.Now(),
		Sex:       sex2,
		Job:       job1,
		Insurance: insurance1,
		UserNurse: nurse1,
	}
	db.Model(&Patient{}).Create(&patient2)

	patient3 := Patient{
		Firstname: "สมชาย",
		Lastname:  "มาอุดฟัน",
		Age:       57,
		IDcard:    "1329900000005",
		Tel:       "0934547915",
		Time:      time.Now(),
		Sex:       sex2,
		Job:       job1,
		Insurance: insurance1,
		UserNurse: nurse2,
	}
	db.Model(&Patient{}).Create(&patient3)

	// ยาและเวชภัณฑ์ ------------------------------------------------------------------------------------------------
	MedicalProduct1 := MedicalProduct{
		Name: "Paracetamol(กระปุก)",
	}
	db.Model(&MedicalProduct{}).Create(&MedicalProduct1)

	MedicalProduct2 := MedicalProduct{
		Name: "Paracetamol(เม็ด)",
	}
	db.Model(&MedicalProduct{}).Create(&MedicalProduct2)

	MedicalProduct3 := MedicalProduct{
		Name: "ไหมขัดฟัน",
	}
	db.Model(&MedicalProduct{}).Create(&MedicalProduct3)

	// คัดกรองผู้ป่วย ------------------------------------------------------------------------------------------------
	screening1 := Screening{
		Illnesses:      "ปวดฟัน",
		Detail:         "ปวดฟันมานาน 1 ชั่วโมง",
		Queue:          "A10",
		Patient:        patient1,
		UserDentistass: dentistass1,
		MedicalProduct: MedicalProduct2,
	}
	db.Model(&Screening{}).Create(&screening1)

	screening2 := Screening{
		Illnesses:      "เหงือกอักเสบ",
		Detail:         "มีอาการเหงือกบวม",
		Queue:          "A11",
		Patient:        patient2,
		UserDentistass: dentistass1,
		MedicalProduct: MedicalProduct1,
	}
	db.Model(&Screening{}).Create(&screening2)

	screening3 := Screening{
		Illnesses:      "ปวดฟัน",
		Detail:         "ปวดฟันมานาน 2 ชั่วโมง",
		Queue:          "A12",
		Patient:        patient3,
		UserDentistass: dentistass1,
		MedicalProduct: MedicalProduct2,
	}
	db.Model(&Screening{}).Create(&screening3)

	// ใบวินิจัย ------------------------------------------------------------------------------------------------------------------
	treatment1 := Treatment{
		PrescriptionRaw:  "A12",
		PrescriptionNote: "",
		ToothNumber:      "21",
		Date:             time.Now(),
		Screening:        screening1,
		UserDentist:      dentist1,
		RemedyType:       remedy1,
	}
	db.Model(&Treatment{}).Create(&treatment1)

	treatment2 := Treatment{
		PrescriptionRaw:  "A12",
		PrescriptionNote: "",
		ToothNumber:      "21",
		Date:             time.Now(),
		Screening:        screening2,
		UserDentist:      dentist1,
		RemedyType:       remedy2,
	}
	db.Model(&Treatment{}).Create(&treatment2)

	treatment3 := Treatment{
		PrescriptionRaw:  "A12",
		PrescriptionNote: "",
		ToothNumber:      "21",
		Date:             time.Now(),
		Screening:        screening3,
		UserDentist:      dentist2,
		RemedyType:       remedy3,
	}
	db.Model(&Treatment{}).Create(&treatment3)

	// การนัดหมาย -------------------------------------------------------------------------------------------------------------
	appoint1 := Appoint{
		AppointTime: time.Now(),
		Todo:        "งดน้ำ 3 ชั่วโมง",
		UserDentist: dentist1,
		Patient:     patient1,
		RemedyType:  remedy1,
	}
	db.Model(&Appoint{}).Create(&appoint1)
	appoint2 := Appoint{
		AppointTime: time.Now(),
		Todo:        "-",
		UserDentist: dentist1,
		Patient:     patient2,
		RemedyType:  remedy2,
	}
	db.Model(&Appoint{}).Create(&appoint2)

	appoint3 := Appoint{
		AppointTime: time.Now(),
		Todo:        "งดอาหาร 12 ชั่วโมง",
		UserDentist: dentist1,
		Patient:     patient1,
		RemedyType:  remedy3,
	}
	db.Model(&Appoint{}).Create(&appoint3)

	// รายการบันทึกการจ่ายยา ----------------------------------------------------------------------------------------------------
	MedRecord1 := MedRecord{
		Amount:         2,
		Treatment:      treatment1,
		UserPharmacist: pharmacist1,
		MedicalProduct: MedicalProduct2,
	}
	db.Model(&MedRecord{}).Create(&MedRecord1)

	MedRecord2 := MedRecord{
		Amount:         2,
		Treatment:      treatment2,
		UserPharmacist: pharmacist1,
		MedicalProduct: MedicalProduct1,
	}
	db.Model(&MedRecord{}).Create(&MedRecord2)

	MedRecord3 := MedRecord{
		Amount:         3,
		Treatment:      treatment3,
		UserPharmacist: pharmacist2,
		MedicalProduct: MedicalProduct3,
	}
	db.Model(&MedRecord{}).Create(&MedRecord3)

	// การชำระเงิน ------------------------------------------------------------------------------------------
	Payment1 := Payment{
		Price:         2000.00,
		Paytime:       time.Now(),
		Note:          "",
		Patient:       patient1,
		UserFinancial: financial1,
		RemedyType:    remedy1,
	}
	db.Model(&Payment{}).Create(&Payment1)

	Payment2 := Payment{
		Price:         200.00,
		Paytime:       time.Now(),
		Note:          "",
		Patient:       patient2,
		UserFinancial: financial1,
		RemedyType:    remedy2,
	}
	db.Model(&Payment{}).Create(&Payment2)

	Payment3 := Payment{
		Price:         500.00,
		Paytime:       time.Now(),
		Note:          "",
		Patient:       patient3,
		UserFinancial: financial1,
		RemedyType:    remedy3,
	}
	db.Model(&Payment{}).Create(&Payment3)

}
```

./main.go

```
package main

import (
	"github.com/ApisitSamorod/SA62G12/controller"
	"github.com/ApisitSamorod/SA62G12/entity"

	"github.com/gin-gonic/gin"
)

func main() {

	entity.SetupDatabase()

	r := gin.Default()
	r.Use(CORSMiddleware())

	// Appoint
	r.GET("/appoints", controller.ListAppoint)
	r.POST("/appoint", controller.CreateAppoint)

	// Insurance
	r.GET("/insrs", controller.ListInsurance)
	r.POST("/insr", controller.CreateInsurance)

	// Job
	r.GET("/jobs", controller.ListJob)

	r.POST("/job", controller.CreateJob)

	// MedicalProduct
	r.GET("/medical_products", controller.ListMedicalProduct)
	r.POST("/medical_product", controller.CreateMedicalProduct)

	// MedRecord
	r.GET("/api/MedRec", controller.ListMedRecord)
	r.POST("/api/submit", controller.CreateMedRecord)

	// Patient
	r.GET("/patients", controller.ListPatient)
	r.POST("/patient", controller.CreatePatient)

	// RemedyType
	r.GET("/remedy_types", controller.ListRemedyType)
	r.POST("/remedy_type", controller.CreateRemedyType)

	// Role
	r.GET("/roles", controller.ListRole)
	r.POST("/role", controller.CreateRole)

	// Screening
	r.GET("/screenings", controller.ListScreening)
	r.POST("/screening", controller.CreateScreening)

	// Sex
	r.GET("/sexs", controller.ListSex)
	r.POST("/sex", controller.CreateSex)

	// Treatment
	r.POST("/treatmentRecord", controller.CreateTreatment)
	r.GET("/treatmentRecords", controller.ListTreatmentRecord)

	// User
	r.GET("/users", controller.ListUser)
	r.GET("/user/dentist/:id", controller.GetUserDentist)
	r.GET("/user/dentistass", controller.GetUserDentistass)
	r.GET("/user/nurse", controller.GetUserNurse)
	r.GET("/user/pharmacist", controller.GetUserPharmacist)
	r.GET("/user/financial", controller.GetUserFinancial)
	r.POST("/user", controller.CreateUser)
```

```
      // Authentication Routes
      r.POST("/login", controller.Login)

      // Run the server
      r.POST("/TRMlogin", controller.TRM_LoginToSite)
      r.GET("/TRMlogout", controller.TRM_Logout)
      r.GET("/TRMauth", controller.TRM_CheckAuth)

      r.Run()
}

func CORSMiddleware() gin.HandlerFunc {

      return func(c *gin.Context) {

            c.Writer.Header().Set("Access-Control-Allow-Origin",
"http://localhost:3000")

            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()

      }

}
```