GROUP-6 PS3: Documentation

Url: https://web06.cs.ait.ac.th/
Gitlab repo: https://gitlab.com/ait-fsad-2022/web6/PS1

Project: Restaurant table ticketing and menu ordering system using QR codes

1. User profiles

Profile 1

Name: Thammanit Nimitwanitch
Age: 37
Occupation: Restaurant Owner
Marital status: Married
Housing situation: Home
Income: $23,076
Relevant short term goal:Improve efficiency of restaurant service, accommodate to a larger crowd, food safety
Relevant Long term goal: Expand restaurant to many areas in Thailand
Questions:

- Do I need it? Will it help my business processes?
- Is the recurring financial cost worth it?
- Is training employees to use the system worth it?

Technology used: PC, Iphone, fiber
Other information:

As a busy person, Thammanit doesn’t like wasting time. As owner of a very busy restaurant, she wants to be able to keep customer satisfaction high and minimize downtime from one customer to another.

Profile 2

Name: Paan Kitjakarn
Age: 27
Occupation: Cashier
Marital status: Single
Housing situation: Apartment
Income: $11,158
Relevant Short term goal: Increase customer satisfaction, get a raise, be recognized at work
Relevant Long term goal: Gather experience and technical fluency of software used at work for her own restaurant in the future
Questions:

- Is the app easy to learn and use?
- Is it responsive and fast?
- Can I quickly find information I need with it? Faster than pen and paper?
- Would it be interesting to use that app for myself?

Technology used: PC, Iphone, fiber
Other information:
Paan has experience working in restaurants. Her previous work experience didn’t involve using any technology and using pen and paper for everything, in a fast-paced environment that is error prone and knows too well customer satisfaction is very important.

Profile 3

Name: Rak Mahagitsiri
Age: 24
Occupation: Student
Marital status: Single
Housing situation: University Dorm
Income: None
Relevant Short term goal: Wait less for queueing at restaurant, get more useful information from queues to avoid sitting out for hours
Relevant Long term goal: Hassle-free eating, one app/service for every place
Questions:

- Is it easy to use? Is it fast?
- Are registration processes annoying?
- Why would I use this app instead of a normal paper menu or queue ticket?

Technology used: PC, Iphone, fiber
Other information:
Rak likes to go out often and tries to avoid queues at restaurants so that he doesn't waste too much time waiting and coming back to finish his class labs.
He uses an old iPhone that runs out of storage quick because of his numerous photos and doesn't like to download apps that he'll use rarely.

2. Offline & Online Alternatives

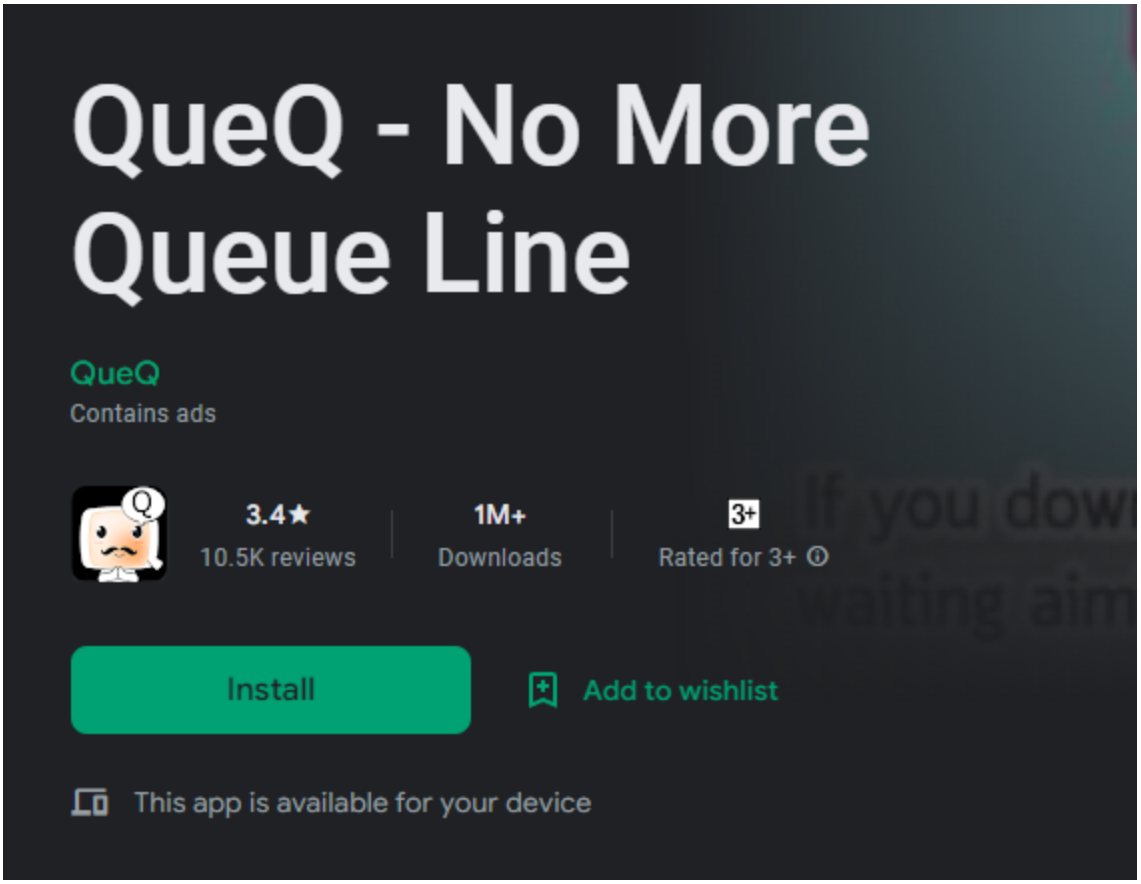
Offline Alternatives

As we observed, many restaurants still use the "pen and paper" way, sometimes you get given a paper ticket of your number (or not at all, you given an estimate waiting time) while waiting for a spot in a popular and busy restaurant, and you either get called out loud directly or your number will show on screen.
You are unsure how long you will wait, 10mins to almost an hour and that you can't walk away too far otherwise you may miss your spot.
Having a way to track when it's your turn and receive a live notification of when you get a spot or close to get a spot may be very helpful.
In the restaurant perspective, they might be able to track more accurately waiting times and downtimes if they use a IT system.

Online Alternatives

These are the top results from Play Store and Google Search in Thailand, and for some of them we already tried them in real life.

QueQ



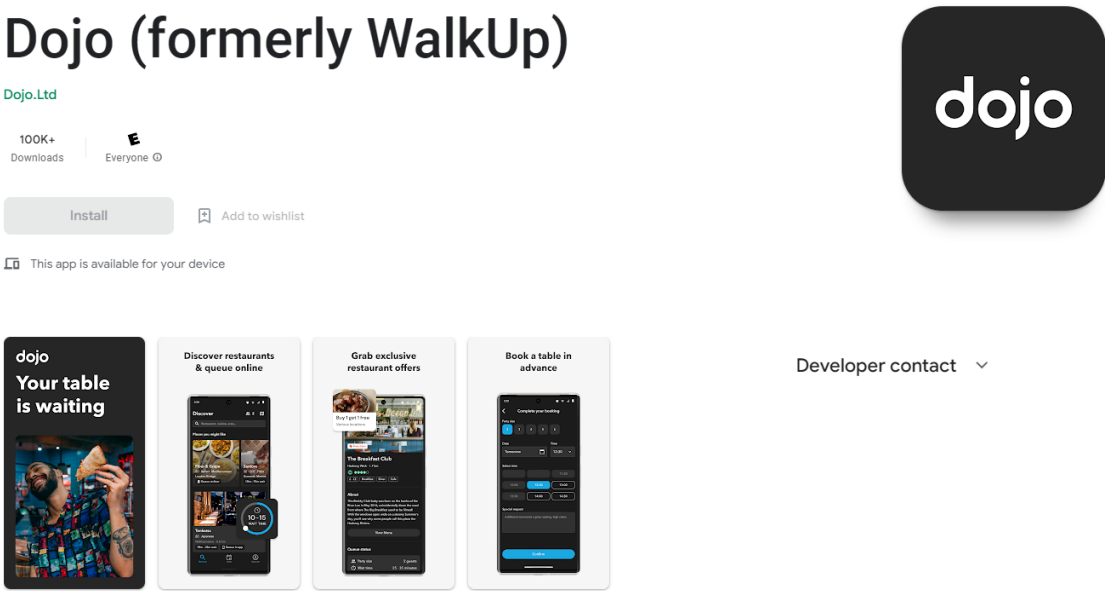
Advantages:

- **Easy and minimal interface**
- **In-app promotions**

Cons:

- **Many bugs**
- **First onboarding can be annoying**
- **Lacks some features**
- **Notifications can bug and not show up**

Dojo



Advantages:

- **Full solution suite, find restaurants nearby and queue**
- **Great interface**

Cons:

- **Not catered to Thailand**
- **We need a simpler solution**

3. Hosting and DNS

Domain name choice

Our product will start servicing specific restaurants in Thailand and so will use the extension **.co.th** (example shopee.co.th), which is a second-level extension restricted for use in legal businesses in Thailand. For a name like **yumQ**, we could use **yumq.co.th**

DNS provider

All DNS providers follow the same process, after the initial payment we get access to their dashboard from where you can update DNS records. In some cases, hosting providers also provide domains (such as godaddy) but in our case, our DNS provider and hosting provider will be different, with one provided by a Thai website and the other hosted by AIT. In that case we need to grab the hostname from our server and update our nameservers in the domain

registrar's control panel. From there, the domain will be resolved automatically towards our server. It will take a while to update, as DNS propagation takes a few hours up to 48 hours.

Below are some domain name registrars offering the extension **.co.th:**

Name	**Price**	**Advantages**	**Cons**
Thaidomainhosting.com	฿ 900.00/year	-Local thai solution -Offer email services too -In service for 20 years -Support accessible on LINE	
thdomain.thnic.co.th	฿ 800.00/year	- Cheaper	- Don't look like many services provided
netim.com	$48.00 /year	-Full hosting solution with DNS -Free for 1 website	-Slow as servers in france -Expensive

Alternatively, we could also use a more global extension **.com**. For our project purposes we will look at shared servers only until we need to accommodate more traffic.

Name	**Price**	**Advantages**	**Cons**
namecheap.com	Up to $13.98/yr for .com domains 2.18$ per months for shared server	- One of the cheapest and known hosting provider, reliable - All in one solution providing various services - Great user interface - Email free with hosting - Free custom nameservers	- Less discounts than most other providers - No phone support
godaddy.com	Up to $13.39/yr, 22$ for 2 years 4$ per month for shared server	- Most popular and cheap hosting provider with good deals - Great discounting on one year most of the time	- Doesn't have a good reputation in security and customer services because of its past - Email hosting pricing separate and lot of upselling for other services

Other hosting services include: inMotion, HostGator.

Other Hosting Providers

1. **AIT Servers**

AIT servers is the default solution for now, providing free hosting for our application and more "DIY" configuration options.

Advantages:

- Free
- Easy access via the campus

Cons:

- Often down and require frequent backups
- Proxy can be a hassle to set up at first
- We need to do sysadmin work as well

2. **Heroku**

Heroku is a cloud-as-service platform that enables developers to easily deploy applications for free without the hassle of server configuration. Owned by Salesforce, it will end its free tier usage at the end of November 2022.

Advantages:

- Easy deployment of various applications using any framework such as Rails
- Most backend tools are provided if needed including Postgresql, Cron jobs, Backups, continuous integration and deployment with Github
- Focus on developing and deploying

Cons:

- Not free anymore on the 28th November

3. **Digital Ocean**

Digitalocean offers a more DIY solution than other hosting solutions as we need to start our own Linux droplet and do the same server setup that we did for AIT server.

Advantages:

- More freedom, DIY solution on server setup (good or bad)
- Many discounts to get started including from Github Education

Cons:

- Must be willing to spend efforts in sysadmin
- Can be more expensive than other solutions if not done properly

Moving to a more permanent server

This is a fairly easy task if the hosting service already provides continuous deployment service for rails applications like Heroku, otherwise redo the server configuration setups will be required.

Database can be dumped and imported easily into a new Postgresql server

4. Intellectual Property

All title to and the rights in the Services, including any and all technology, software, and content, including ownership rights to patents (registrations, renewals, and pending applications), copyrights, trademarks, trade secrets, the Company's or third-party hardware, other technology is exclusive property of the company and developers.
Trademarks, graphics such as logos are trademarks are the trademarks of their respective owners

5. Database Backups

Required scripts to do the backup of the postgresql database.

1. **pg_backup.config** - The main configuration file. This should be the only file which needs user modifications.
2. **pg_backup.sh** - The normal backup script which will go through each database and save a gzipped and/or a custom format copy of the backup into a date-based directory.
3. **pg_backup_rotated.sh** - The same as above except it will delete expired backups based on the configuration.

   pg_backup.config

   ```
   ################################
   ## POSTGRESQL BACKUP CONFIG ##
   ################################

   # Optional system user to run backups as.  If the user the script is running as doesn't match this
   # the script terminates.  Leave blank to skip check.
   BACKUP_USER=

   # Optional hostname to adhere to pg_hba policies.  Will default to "localhost" if none specified.
   HOSTNAME=

   # Optional username to connect to database as.  Will default to "postgres" if none specified.
   USERNAME=

   # This dir will be created if it doesn't exist.  This must be writable by the user the script is
   # running as.
   BACKUP_DIR=/home/backups/database/postgresql/
   ```

```
# List of strings to match against in database name, separated by space or comma, for which we only
# wish to keep a backup of the schema, not the data. Any database names which contain any of these
# values will be considered candidates. (e.g. "system_log" will match "dev_system_log_2010-01")
SCHEMA_ONLY_LIST=""

# Will produce a custom-format backup if set to "yes"
ENABLE_CUSTOM_BACKUPS=yes

# Will produce a gzipped plain-format backup if set to "yes"
ENABLE_PLAIN_BACKUPS=yes

# Will produce gzipped sql file containing the cluster globals, like users and passwords, if set to "yes"
ENABLE_GLOBALS_BACKUPS=yes


#### SETTINGS FOR ROTATED BACKUPS ####

# Which day to take the weekly backup from (1-7 = Monday-Sunday)
DAY_OF_WEEK_TO_KEEP=7

# Number of days to keep daily backups
DAYS_TO_KEEP=7

# How many weeks to keep weekly backups
WEEKS_TO_KEEP=1

########################################
```

pg_backup.sh

```
#!/bin/bash

#############################
####### LOAD CONFIG #######
#############################

while [ $# -gt 0 ]; do
```

```
        case $1 in
                -c)
                        if [ -r "$2" ]; then
                                source "$2"
                                shift 2
                        else
                                ${ECHO} "Unreadable config file \"$2\"" 1>&2
                                exit 1
                        fi
                        ;;
                *)
                        ${ECHO} "Unknown Option \"$1\"" 1>&2
                        exit 2
                        ;;
        esac
done

if [ $# = 0 ]; then
        SCRIPTPATH=$(cd ${0%/*} && pwd -P)
        source $SCRIPTPATH/pg_backup.config
fi;

##############################
#### PRE-BACKUP CHECKS ####
##############################

# Make sure we're running as the required backup user
if [ "$BACKUP_USER" != "" -a "$(id -un)" != "$BACKUP_USER" ]; then
        echo "This script must be run as $BACKUP_USER. Exiting." 1>&2
        exit 1;
fi;


##############################
### INITIALISE DEFAULTS ###
##############################

if [ ! $HOSTNAME ]; then
        HOSTNAME="localhost"
fi;

if [ ! $USERNAME ]; then
```

```
        USERNAME="postgres"
fi;


#############################
#### START THE BACKUPS ####
#############################


FINAL_BACKUP_DIR=$BACKUP_DIR"`date +\%Y-\%m-\%d`/"

echo "Making backup directory in $FINAL_BACKUP_DIR"

if ! mkdir -p $FINAL_BACKUP_DIR; then
        echo "Cannot create backup directory in $FINAL_BACKUP_DIR. Go and fix it!"
1>&2
        exit 1;
fi;


########################
### GLOBALS BACKUPS ###
########################

echo -e "\n\nPerforming globals backup"
echo -e "--------------------------------------------\n"

if [ $ENABLE_GLOBALS_BACKUPS = "yes" ]
then
       echo "Globals backup"

       set -o pipefail
       if ! pg_dumpall -g -h "$HOSTNAME" -U "$USERNAME" | gzip >
$FINAL_BACKUP_DIR"globals".sql.gz.in_progress; then
              echo "[!!ERROR!!] Failed to produce globals backup" 1>&2
       else
              mv $FINAL_BACKUP_DIR"globals".sql.gz.in_progress
$FINAL_BACKUP_DIR"globals".sql.gz
       fi
       set +o pipefail
else
        echo "None"
```

```
fi


############################
### SCHEMA-ONLY BACKUPS ###
############################

for SCHEMA_ONLY_DB in ${SCHEMA_ONLY_LIST//,/ }
do
        SCHEMA_ONLY_CLAUSE="$SCHEMA_ONLY_CLAUSE or datname ~
'$SCHEMA_ONLY_DB'"
done

SCHEMA_ONLY_QUERY="select datname from pg_database where false
$SCHEMA_ONLY_CLAUSE order by datname;"

echo -e "\n\nPerforming schema-only backups"
echo -e "--------------------------------------------\n"

SCHEMA_ONLY_DB_LIST=`psql -h "$HOSTNAME" -U "$USERNAME" -At -c
"$SCHEMA_ONLY_QUERY" postgres`

echo -e "The following databases were matched for schema-only
backup:\n${SCHEMA_ONLY_DB_LIST}\n"

for DATABASE in $SCHEMA_ONLY_DB_LIST
do
        echo "Schema-only backup of $DATABASE"

        set -o pipefail
        if ! pg_dump -Fp -s -h "$HOSTNAME" -U "$USERNAME" "$DATABASE" | gzip >
$FINAL_BACKUP_DIR"$DATABASE"_SCHEMA.sql.gz.in_progress; then
                echo "[!!ERROR!!] Failed to backup database schema of $DATABASE"
1>&2
        else
                mv $FINAL_BACKUP_DIR"$DATABASE"_SCHEMA.sql.gz.in_progress
$FINAL_BACKUP_DIR"$DATABASE"_SCHEMA.sql.gz
        fi
        set +o pipefail
done
```

```
############################
###### FULL BACKUPS #######
############################

for SCHEMA_ONLY_DB in ${SCHEMA_ONLY_LIST//,/ }
do

EXCLUDE_SCHEMA_ONLY_CLAUSE="$EXCLUDE_SCHEMA_ONLY_CLAUSE and
datname !~ '$SCHEMA_ONLY_DB'"
done

FULL_BACKUP_QUERY="select datname from pg_database where not datistemplate
and datallowconn $EXCLUDE_SCHEMA_ONLY_CLAUSE order by datname;"

echo -e "\n\nPerforming full backups"
echo -e "----------------------------------------\n"

for DATABASE in `psql -h "$HOSTNAME" -U "$USERNAME" -At -c
"$FULL_BACKUP_QUERY" postgres`
do
        if [ $ENABLE_PLAIN_BACKUPS = "yes" ]
        then
                echo "Plain backup of $DATABASE"

                set -o pipefail
                if ! pg_dump -Fp -h "$HOSTNAME" -U "$USERNAME" "$DATABASE" |
gzip > $FINAL_BACKUP_DIR"$DATABASE".sql.gz.in_progress; then
                        echo "[!!ERROR!!] Failed to produce plain backup database
$DATABASE" 1>&2
                else
                        mv $FINAL_BACKUP_DIR"$DATABASE".sql.gz.in_progress
$FINAL_BACKUP_DIR"$DATABASE".sql.gz
                fi
                set +o pipefail
        fi

        if [ $ENABLE_CUSTOM_BACKUPS = "yes" ]
        then
                echo "Custom backup of $DATABASE"

                if ! pg_dump -Fc -h "$HOSTNAME" -U "$USERNAME" "$DATABASE" -f
$FINAL_BACKUP_DIR"$DATABASE".custom.in_progress; then
```

```
                        echo "[!!ERROR!!] Failed to produce custom backup database
$DATABASE" 1>&2
                else
                        mv $FINAL_BACKUP_DIR"$DATABASE".custom.in_progress
$FINAL_BACKUP_DIR"$DATABASE".custom
                fi
        fi

done

echo -e "\nAll database backups complete!"
```

pg_backup_rotated.sh

```
#!/bin/bash

###############################
####### LOAD CONFIG #######
###############################

while [ $# -gt 0 ]; do
        case $1 in
                -c)
                        CONFIG_FILE_PATH="$2"
                        shift 2
                        ;;
                *)
                        ${ECHO} "Unknown Option \"$1\"" 1>&2
                        exit 2
                        ;;
        esac
done

if [ -z $CONFIG_FILE_PATH ] ; then
        SCRIPTPATH=$(cd ${0%/*} && pwd -P)
        CONFIG_FILE_PATH="${SCRIPTPATH}/pg_backup.config"
fi

if [ ! -r ${CONFIG_FILE_PATH} ] ; then
        echo "Could not load config file from ${CONFIG_FILE_PATH}" 1>&2
        exit 1
```

```
fi

source "${CONFIG_FILE_PATH}"

#############################
#### PRE-BACKUP CHECKS ####
#############################

# Make sure we're running as the required backup user
if [ "$BACKUP_USER" != "" -a "$(id -un)" != "$BACKUP_USER" ] ; then
        echo "This script must be run as $BACKUP_USER. Exiting." 1>&2
        exit 1
fi


#############################
### INITIALISE DEFAULTS ###
#############################

if [ ! $HOSTNAME ]; then
        HOSTNAME="localhost"
fi;

if [ ! $USERNAME ]; then
        USERNAME="postgres"
fi;


#############################
#### START THE BACKUPS ####
#############################

function perform_backups()
{
        SUFFIX=$1
        FINAL_BACKUP_DIR=$BACKUP_DIR"`date +\%Y-\%m-\%d`$SUFFIX/"

        echo "Making backup directory in $FINAL_BACKUP_DIR"

        if ! mkdir -p $FINAL_BACKUP_DIR; then
                echo "Cannot create backup directory in $FINAL_BACKUP_DIR. Go and
fix it!" 1>&2
```

```
                exit 1;
        fi;

        ##########################
        ### GLOBALS BACKUPS ###
        ##########################

        echo -e "\n\nPerforming globals backup"
        echo -e "----------------------------------------\n"

        if [ $ENABLE_GLOBALS_BACKUPS = "yes" ]
        then
                echo "Globals backup"

                set -o pipefail
                if ! pg_dumpall -g -h "$HOSTNAME" -U "$USERNAME" | gzip >
$FINAL_BACKUP_DIR"globals".sql.gz.in_progress; then
                        echo "[!!ERROR!!] Failed to produce globals backup" 1>&2
                else
                        mv $FINAL_BACKUP_DIR"globals".sql.gz.in_progress
$FINAL_BACKUP_DIR"globals".sql.gz
                fi
                set +o pipefail
        else
                echo "None"
        fi


        ############################
        ### SCHEMA-ONLY BACKUPS ###
        ############################

        for SCHEMA_ONLY_DB in ${SCHEMA_ONLY_LIST//,/ }
        do
                SCHEMA_ONLY_CLAUSE="$SCHEMA_ONLY_CLAUSE or datname ~
'$SCHEMA_ONLY_DB'"
        done

        SCHEMA_ONLY_QUERY="select datname from pg_database where false
$SCHEMA_ONLY_CLAUSE order by datname;"

        echo -e "\n\nPerforming schema-only backups"
```

```
        echo -e "----------------------------------------\n"

        SCHEMA_ONLY_DB_LIST=`psql -h "$HOSTNAME" -U "$USERNAME" -At -c
"$SCHEMA_ONLY_QUERY" postgres`

        echo -e "The following databases were matched for schema-only
backup:\n${SCHEMA_ONLY_DB_LIST}\n"

        for DATABASE in $SCHEMA_ONLY_DB_LIST
        do
                echo "Schema-only backup of $DATABASE"
                  set -o pipefail
                if ! pg_dump -Fp -s -h "$HOSTNAME" -U "$USERNAME" "$DATABASE" |
gzip > $FINAL_BACKUP_DIR"$DATABASE"_SCHEMA.sql.gz.in_progress; then
                        echo "[!!ERROR!!] Failed to backup database schema of $DATABASE"
1>&2
                else
                        mv
$FINAL_BACKUP_DIR"$DATABASE"_SCHEMA.sql.gz.in_progress
$FINAL_BACKUP_DIR"$DATABASE"_SCHEMA.sql.gz
                fi
                set +o pipefail
        done



        ###############################
        ###### FULL BACKUPS #######
        ###############################

        for SCHEMA_ONLY_DB in ${SCHEMA_ONLY_LIST//,/ }
        do


EXCLUDE_SCHEMA_ONLY_CLAUSE="$EXCLUDE_SCHEMA_ONLY_CLAUSE and
datname !~ '$SCHEMA_ONLY_DB'"
        done

        FULL_BACKUP_QUERY="select datname from pg_database where not
datistemplate and datallowconn $EXCLUDE_SCHEMA_ONLY_CLAUSE order by
datname;"

        echo -e "\n\nPerforming full backups"
```

```
        echo -e "----------------------------------------\n"

        for DATABASE in `psql -h "$HOSTNAME" -U "$USERNAME" -At -c
"$FULL_BACKUP_QUERY" postgres`
        do
                if [ $ENABLE_PLAIN_BACKUPS = "yes" ]
                then
                        echo "Plain backup of $DATABASE"

                        set -o pipefail
                        if ! pg_dump -Fp -h "$HOSTNAME" -U "$USERNAME"
"$DATABASE" | gzip > $FINAL_BACKUP_DIR"$DATABASE".sql.gz.in_progress; then
                                echo "[!!ERROR!!] Failed to produce plain backup database
$DATABASE" 1>&2
                        else
                                mv $FINAL_BACKUP_DIR"$DATABASE".sql.gz.in_progress
$FINAL_BACKUP_DIR"$DATABASE".sql.gz
                        fi
                        set +o pipefail

                fi

                if [ $ENABLE_CUSTOM_BACKUPS = "yes" ]
                then
                        echo "Custom backup of $DATABASE"

                        if ! pg_dump -Fc -h "$HOSTNAME" -U "$USERNAME"
"$DATABASE" -f $FINAL_BACKUP_DIR"$DATABASE".custom.in_progress; then
                                echo "[!!ERROR!!] Failed to produce custom backup database
$DATABASE"
                        else
                                mv $FINAL_BACKUP_DIR"$DATABASE".custom.in_progress
$FINAL_BACKUP_DIR"$DATABASE".custom
                        fi
                fi

        done

        echo -e "\nAll database backups complete!"
}

# MONTHLY BACKUPS
```

```
DAY_OF_MONTH=`date +%d`

if [ $DAY_OF_MONTH -eq 1 ];
then
        # Delete all expired monthly directories
        find $BACKUP_DIR -maxdepth 1 -name "*-monthly" -exec rm -rf '{}' ';'

        perform_backups "-monthly"

        exit 0;
fi

# WEEKLY BACKUPS

DAY_OF_WEEK=`date +%u` #1-7 (Monday-Sunday)
EXPIRED_DAYS=`expr $((($WEEKS_TO_KEEP * 7) + 1))`

if [ $DAY_OF_WEEK = $DAY_OF_WEEK_TO_KEEP ];
then
        # Delete all expired weekly directories
        find $BACKUP_DIR -maxdepth 1 -mtime +$EXPIRED_DAYS -name "*-weekly" -
exec rm -rf '{}' ';'

        perform_backups "-weekly"

        exit 0;
fi

# DAILY BACKUPS

# Delete daily backups 7 days old or more
find $BACKUP_DIR -maxdepth 1 -mtime +$DAYS_TO_KEEP -name "*-daily" -exec rm
-rf '{}' ';'

perform_backups "-daily"
```

Automation with the cron

After creating the scripts, now its time to do the nightly dump using cron.

Create daily cron jobs in /etc/crontab. The backup runs daily at 12 am and the cleaning of the database backups job runs daily at 4am.

```
# m h dom mon dow user  command
0 1 * * * postgres /var/scripts/pg_backup.sh #location to be changed
0 5 * * * postgres /var/scripts/pg_backup_rotated.sh
```

6. CI/CD

For the GitlabCI to test our application along with PostgreSQL we need a specific setup in the **config/database.yml** file. We use the dotenv-rails gem.

config/database.yml

```
default: &default
  adapter: postgresql
  encoding: unicode
  pool: <%= ENV['DB_POOL'] %>
  username: <%= ENV['DB_USERNAME'] %>
  password: <%= ENV['DB_PASSWORD'] %>
  host: <%= ENV['DB_HOST'] %>
  port: 5432

development:
  <<: *default
  database:

test:
  <<: *default
  database:

production:
  <<: *default
  database:
```

.gitlab-ci.yml

```
stages:
  - build
  - test
```

```
.build:
  image: ruby:2.6.5 #needs to be changed
  cache:
    key: rat-app-name
    paths:
      - apt-cache/
      - vendor/ruby
      - node_modules/
    policy: pull
  before_script:
    - gem install bundler --no-document
    - bundle install --jobs $(nproc) "${FLAGS[@]}" --path=vendor

.db:
  extends: .build
  services:
    - postgres:12.0 #needs to be changed
  variables: #needs to be initialized
    POSTGRES_USER:
    POSTGRES_PASSWORD:
    DB_USERNAME:
    DB_PASSWORD:
    DB_HOST: postgres
    RAILS_ENV: test
    DISABLE_SPRING: 1
    BUNDLE_PATH: vendor/bundle
  before_script:
    - export APT_CACHE_DIR=`pwd`/apt-cache && mkdir -pv $APT_CACHE_DIR
    - apt update -qq && apt -o dir::cache::archives="$APT_CACHE_DIR" install -yqq nodejs
    - gem install bundler --no-document
    - bundle install --jobs $(nproc) "${FLAGS[@]}" --path=vendor
    - bundle exec rails db:create db:schema:load --trace
```

7. Capistrano script

8. References

https://www.globaldata.com/data-insights/macroeconomic/median-household-income-in-thailand/
https://wagecentre.com/work/work-in-asia-and-oceania/salary-in-thailand
https://teleport.org/cities/bangkok/salaries/
https://wiki.postgresql.org/wiki/Automated_Backup_on_Linux
https://www.odoo.com/forum/help-1/how-to-setup-a-regular-postgresql-database-backup-4728
https://hixonrails.com/ruby-on-rails-tutorials/ruby-on-rails-set-up-on-gitlab-with-gitlabci/

Other profile sources
https://qubstudio.com/blog/4-examples-of-ux-personas/
https://www.uxpin.com/studio/blog/persona-examples/
Lead magnets
https://www.searchenginejournal.com/lead-magnet-examples/265245/