Ansible Basic

An Ansible Training Course

[redacted]

DevOps Artisan

8. Advanced Topics

Topics covered

- Ansible Vault
- Roles: file structure, simple example

8.1. Ansible Vault

Ansible Vault

- Some modules require sensitive data to be processed
- This may include private keys, passwords, and more
- To process sensitive data in a secure way, Ansible Vault can be used
- **Ansible Vault** is used to **encrypt** and **decrypt files**
- To manage this process, the **ansible-vault** command is used

Creating an Encrypted File

- To create an encrypted file use:

  -
    ```
    ansible-vault create playbook.yml
    ```

- This command will prompt for a new vault password, and opens the file in **vi** for further editing

- As an alternative for entering password on the prompt, a vault password file may be used, but you’ll have to make sure this file is protected in another way:

  -
    ```
    ansible-vault create –vault-password-file=vault-pass
    playbook.yml
    ```

Creating an Encrypted File - Example

```
student:~$ ansible-vault create secret.yml
New Vault password: 123
Confirm New Vault password: 123
Encryption successful
```

```
student:~$ cat secret.yml
$ANSIBLE_VAULT;1.1;AES256
39656563336538323136363032613366323263613237613333633735623832326631313834643138
303635343430366431613266343962626233633062616666650a626664653636346539623339653631
31363333396435646631623563626132383264343165326635343935633764373735613162613034
616633386138376337 0a32656136643232323639613166633637363734313661623333136613035 61
62643639356139353665346433333663396539363461393862313365333561363663313231376633
3931303634386262643639376233343130363438353334383162
```

Creating an Encrypted File

- To view a vault encrypted file:
  - **`ansible-vault view playbook.yml`**

- To edit:
  - **`ansible-vault edit playbook.yml`**

- Use **`ansible-vault encrypt playbook.yml`** to encrypt an existing file, and use **`ansible-vault decrypt playbook.yml`** to decrypt it

- To change a password on an existing file, use **`ansible-vault rekey`**

Using Playbooks with Vault

- To run a playbook that accesses Vault encrypted files, you need to use **--vault-id @prompt** option to be prompted for a password

- Alternatively, you can store the password as a single-line string in a password file, and access that using the **--vault-password-file=vault-file** option

Include Multiple Vaults

- Until Ansible 2.4 we could include more vault files only if they had the same password.

- Starting with that version, a new option called **`vault-id`** was introduced.
  - This provides the option to include multiple vault files with different passwords.

Include Multiple Vaults - Example

```
student:~$ vi testvault_v2.yml
---
- name: Ansible Vault Playbook
  hosts: hivemaster
  gather_facts: no
  tasks:
  - name: Include var from vault file
    include_vars: "/home/student/secret.yml"

  - name: Include var from another vault file
    include_vars: "/home/student/anothersecret.yml"

  - name: Print var from vault1
    debug:
      msg: "{{ secret_var }}"

  - name: Print var from vault2
    debug:
      msg: "{{ another_secret_var }}"
```

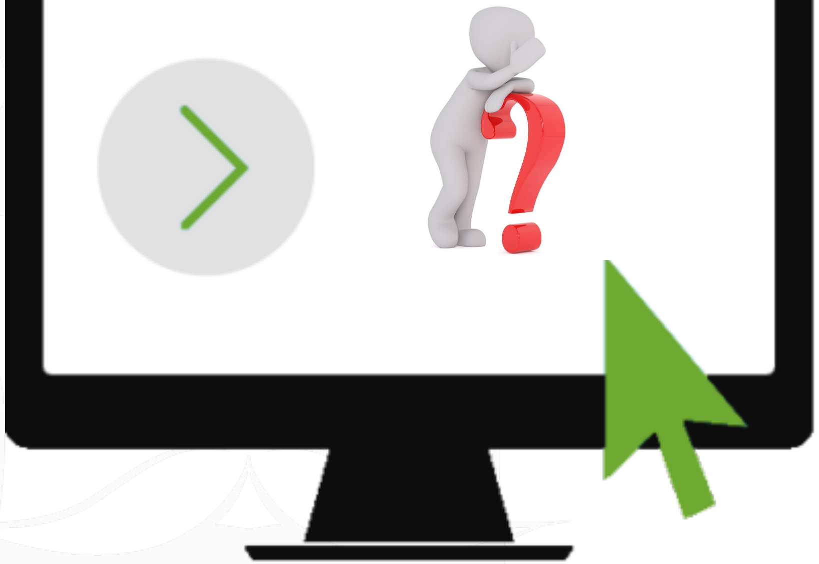
```
student:~$ ansible-playbook testvault_v2.yml --vault-id label1@prompt --vault-id
label2@prompt
```

Managing Vault Files

- When setting up projects with Vault encrypted files, it makes sense to use separate files to store encrypted and non-encrypted variables

- To store host or host-group related variable files, you can use the following structure:

```
| -group_vars
|   |--dbservers
|       |- vars
|       |- vault
```



Q&A

Lab 8: *Ansible Vault*

8.2. Roles

DevOps Artisan

Organizing Ansible Contents

- When working with Ansible, it is recommended to use project directories so that contents can be organized in a consistent way
- Each project directory may have its own ansible.cfg, inventory as well as playbooks
- If the directory grows bigger, variable files and other include files may be used
- And finally, roles can be used to standardize and easily re-use specific parts of Ansible
- For now, consider a role a complete project dedicated to a specific task that is going to be included in the main playbook

Directory Layout – Best Practices

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier
```

What are Roles?

- Ansible Playbooks can be very similar: code used in one playbook can be useful in other playbooks also.

- To make it easy to reuse code, roles can be used.

- A role is a collection of tasks, variables, files, templates and other resources in a fixed directory structure that, as such, can easily be included from a playbook.

What are Roles?

- Roles should be written in a generic way, such that play specifics can be defined as variables in the play, and overwrite the default variables that should be set in the role

- Using Roles makes working with large project more manageable

```
roles/
   common/
     tasks/
     handlers/
     files/
     templates/
     vars/
     defaults/
     meta/
```

Roles Default Structure

- **defaults** contains default values of role variables. If variables are set at the play level as well, these default values are overwritten

- **files** may contain static files that are needed from the role tasks

- **handlers** has a `main.yml` that defines handlers used in the role

- **meta** has a `main.yml` that may be used to include role metadata, such as information about author, license, dependencies and more

Roles Default Structure

- **tasks** contains a `main.yml` that defines the role task definitions
- **templates** is used to store Jinja2 templates
- **tests** may contain an optional inventory file, as well as a `test.yml` playbook that can be used to test the role
- **vars** may contain a `main.yml` with standard variables for the role (which are not meant to be overwritten by playbook variables)

Role Variables

- Variables can be defined at different levels in a role.
- **vars/main.yml** has the role default variables, which are used in default role functioning. They are not intended to be overwritten.
- **defaults/main.yml** can contain default variables. These have a low precedence, and can be overwritten by variables with the same name that are set in the playbook and which have higher precedence.

Role Variables

- Playbook variables will always overwrite the variables as set in the role. Site-specific variables such as secrets and vault encrypted data should always be managed from the playbook, as role variables are intended to be generic

- Role variables are defined in the playbook when calling the role and they have the highest precedence and overwrite playbook variables and well as inventory variables

Role Location

- Roles can be obtained in many ways
  - You can write your own roles
  - For Red Hat Enterprise Linux, the rhel-system-roles package is available
  - The community provides roles through the Ansible Galaxy website
- Roles can be stored at a default location, and from there can easily be used from playbooks
  - **./roles** has highest precedence
  - **~/.ansible/roles** is checked after that
  - **/etc/ansible/roles** is checked next
  - **/usr/share/ansible/** roles is checked last

Roles in a Playbook

- Roles are referred to from playbooks
- Old syntax:

```
- name: role demo
  hosts: all
  roles:
   - role1
   - role2
```

- Newer (recommended) syntax:

```
- hosts: webservers
  tasks:
  - debug:
      msg: "before we run our role"
  - import_role:
      name: example
  - include_role:
      name: example
  - debug:
      msg: "after we ran our role"
```

DevOps Artisan

Role Variables

- When calling a role, role variables can be defined

```
---
- name: role demo
  hosts: all
  roles:
   - role1
   - role2
     var1: cake
     var2: cow
```

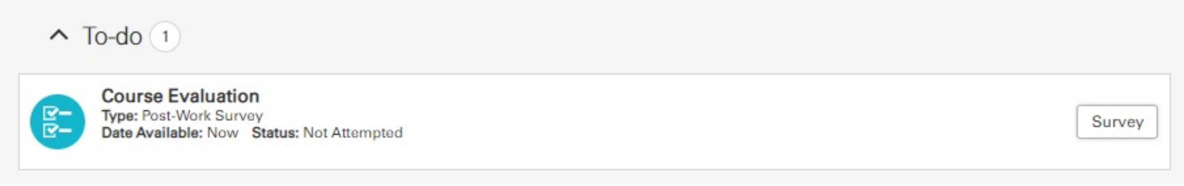
Q&A

Lab 9: *Roles*

- **We Value Your Feedback**

Oracle will soon send you an Online Workshop Evaluation

1. **Your feedback is important!**
2. Measuring the effectiveness of our classes allows us to make continuous improvements to meet your learning needs
3. Login to https://learning.oracle.com and complete the survey

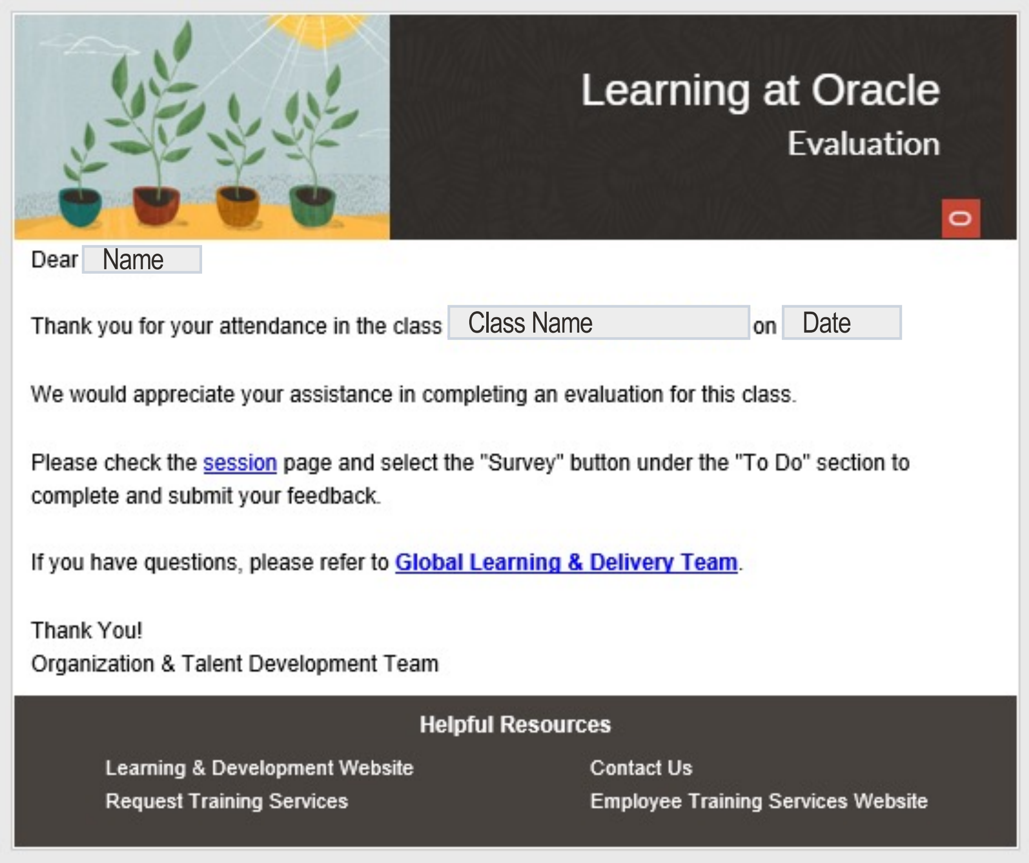


To-do 1

Course Evaluation
Type: Post-Work Survey
Date Available: Now Status: Not Attempted

4. Please follow the prompts (2-3 minutes)
5. Comments are particularly helpful!

With thanks! ☺



Learning at Oracle
Evaluation

Dear Name

Thank you for your attendance in the class Class Name on Date

We would appreciate your assistance in completing an evaluation for this class.

Please check the session page and select the "Survey" button under the "To Do" section to complete and submit your feedback.

If you have questions, please refer to Global Learning & Delivery Team.

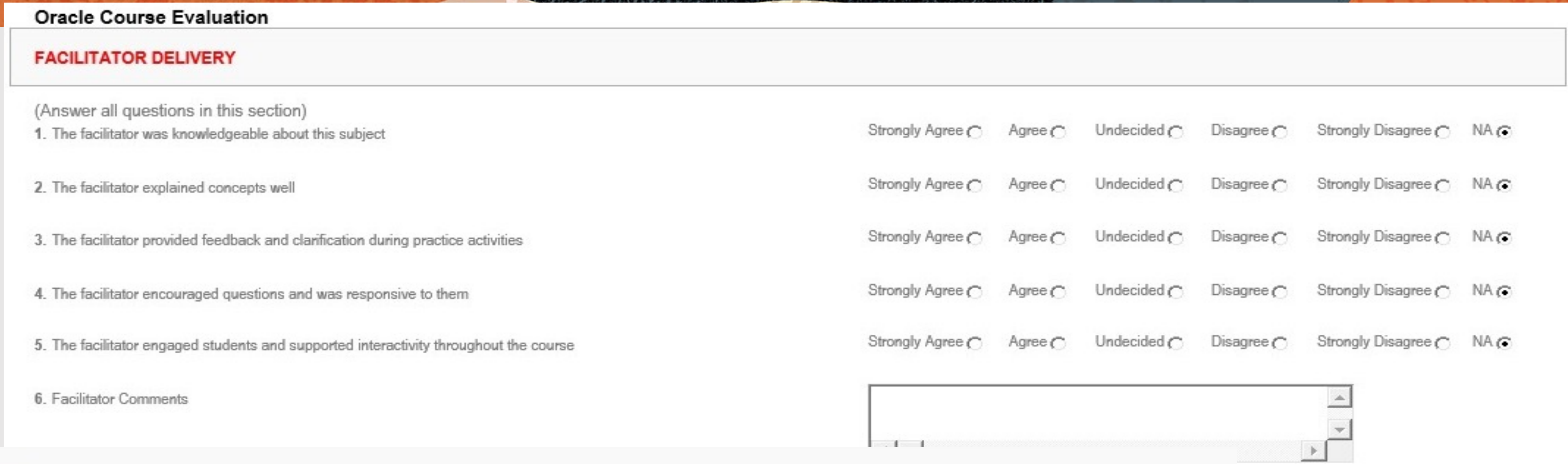
Thank You!
Organization & Talent Development Team

Helpful Resources

Learning & Development Website
Request Training Services
Contact Us
Employee Training Services Website

You will also soon have a follow up request in your Inbox

Oracle Course Evaluation Template



Oracle Course Evaluation

FACILITATOR DELIVERY

(Answer all questions in this section)

1. The facilitator was knowledgeable about this subject — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

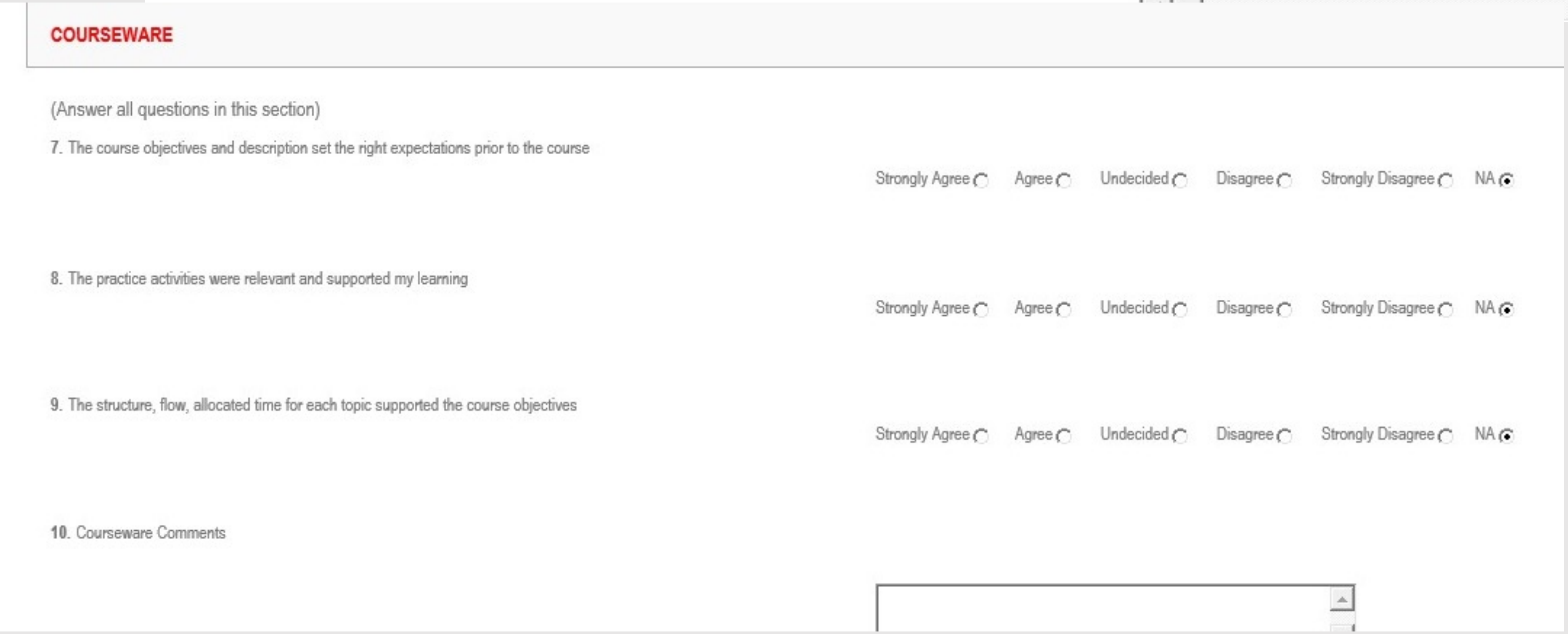
2. The facilitator explained concepts well — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

3. The facilitator provided feedback and clarification during practice activities — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

4. The facilitator encouraged questions and was responsive to them — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

5. The facilitator engaged students and supported interactivity throughout the course — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

6. Facilitator Comments



COURSEWARE

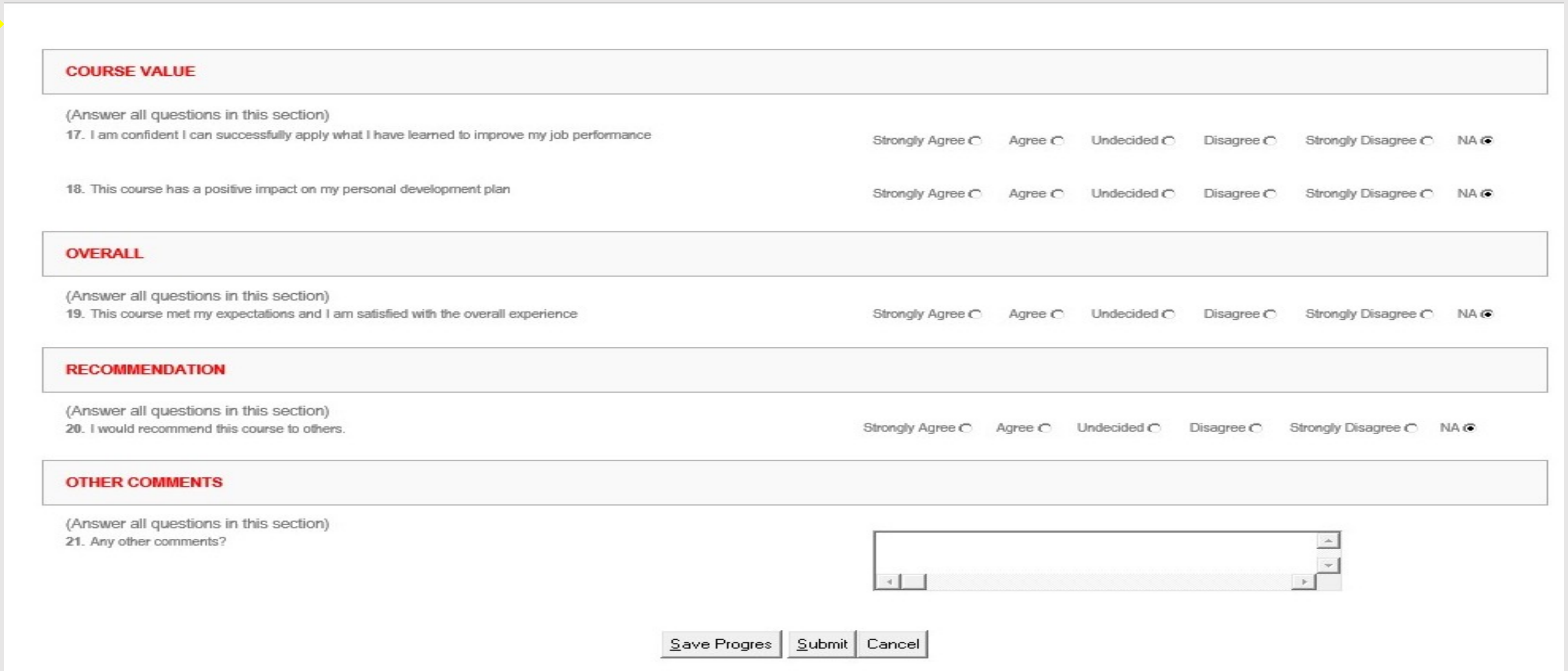
(Answer all questions in this section)

7. The course objectives and description set the right expectations prior to the course — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

8. The practice activities were relevant and supported my learning — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

9. The structure, flow, allocated time for each topic supported the course objectives — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

10. Courseware Comments



COURSE VALUE

(Answer all questions in this section)

17. I am confident I can successfully apply what I have learned to improve my job performance — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

18. This course has a positive impact on my personal development plan — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

OVERALL

(Answer all questions in this section)

19. This course met my expectations and I am satisfied with the overall experience — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

RECOMMENDATION

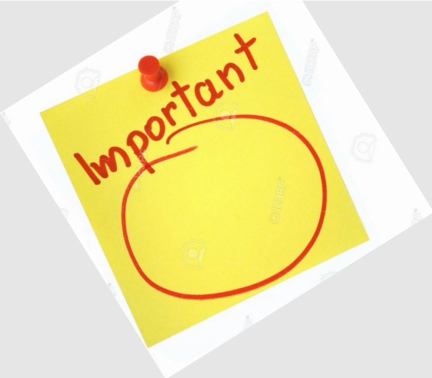
(Answer all questions in this section)

20. I would recommend this course to others. — Strongly Agree ○ Agree ○ Undecided ○ Disagree ○ Strongly Disagree ○ NA ●

OTHER COMMENTS

(Answer all questions in this section)

21. Any other comments?



- ✓ Please make sure to fill in **comments** and give us more hints about your feedback
- ✓ **Thank you** for helping us improve the quality of our service!

More practice, less theory

askformore@devopsartisan.com

DevOps Artisan