

1. Class Name- Application

a. Method - main()

- i. Input : String args[], (*to be written*)
- ii. Output : Runs the application
 1. SpringApplication.run();

2. Class Name- CompanyController

a. Method - showJobSeeker

- i. Input : id, model
- ii. Output : profile of the company
 1. Company company = select a specific company by its id.
 2. Select the model of the company
 3. return "companyprofile"

b. Method - showJob

- i. Input : cid, jobid, model
- ii. Output : job profile
 1. JobPosting p1 = select a specific posted job.
 2. Company company = select a specific company by its id.
 3. Select the model of the job
 4. Select the model of the company
 5. Return "jobprofile"

c. Method - showjobapplications

- i. Input : jobid, model
- ii. Output : job profile
 1. JobPosting p1 = select a specific posted job.
 2. Select the model of the job
 3. Return "jobprofile"

d. Methods - getjobs

- i. Input : companyid, model
- ii. Output : company's jobs
 1. List<> companyJobPostings = new ArrayList<String>();
 2. companyJobPosting = get company's posted jobs in the list
 3. Company company = select a specific company by its id.
 4. Select the model of the jobs
 5. Select the model of the company
 6. Return companyjobs

3. Class name- InterviewController

a. Method - createInterview

- i. Input : appid, location, datetime
- ii. Output : Affirmative reply
 - 1. Jobapplication ja = select a specific job application based on the id
 - 2. ja.setInterviewFlag(true);
 - 3. ja.setInterviewLocation(location);
 - 4. ja.setInterviewTime(Date.valueOf(datetime));
 - 5. ja.setInterviewAccepted(false);
 - 6. jobAppDao.updateApplication(ja);
 - 7. verifyUrl
 - 8. Send affirmative message to the applicant

b. Method - acceptinterview

- i. Input : appid
- ii. Output : Affirmative reply
 - 1. Jobapplication ja = select a specific job application based on the id
 - 2. ja.setInterviewaccepted(true);
 - 3. jobAppDao.updateApplication(ja);
 - 4. Company c = get job lists and companies
 - 5. Send affirmative message to the company

4. Class name - JobapplicationController

a. Method - applyPage

- i. Input : jobseekerId, jobId, Model
- ii. Output : jobapplication

b. Method - Apply

- i. Input : jobseekerId, jobId, Model, file
- ii. Output : userjobprofile, redirect:upload status
 - 1. if (resumeFlag == true)
 - 2. if (file.equals(empty()))
 - 3. redirectAttributes.addFlashAttribute("message", "Please select a file to upload");
 - 4. byte[] bytes = file.get().getBytes();
 - 5. Path path = Paths.get(UPLOADED_FOLDER + file.get().getOriginalFilename());
 - 6. JobApplication ja = new JobApplication();
 - 7. ja = jobAppDao.apply(Integer.parseInt(jobSeekerId), Integer.parseInt(jobId), resumeFlag,
 - 8. JobSeeker js = jobSeekerDao.getJobSeeker(Integer.parseInt(jobSeekerId));
 - 9. JobPosting jp = jobDao.getJobPosting(Integer.parseInt(jobId));
 - 10. emailService.sendSimpleMessage(js.getEmailId(),

```

11. Company company = jp.getCompany();
12. if (ij.contains(Integer.parseInt(jobId)))
13. if (il.contains(Integer.parseInt(jobId)))
14. model.addAttribute("job", jp);
15. model.addAttribute("seeker", js);
16. model.addAttribute("company", company);
17. model.addAttribute("interested", i);
18. model.addAttribute("applied", j);
19. Files.write(path, bytes);
20. JobApplication ja = new JobApplication();
21. ja = jobAppDao.apply(Integer.parseInt(jobSeekerId), Integer.parseInt(jobId),
    resumeFlag, resumePath);
22. JobSeeker js = jobSeekerDao.getJobSeeker(Integer.parseInt(jobSeekerId));
23. JobPosting jp = jobDao.getJobPosting(Integer.parseInt(jobId));
24. emailService.sendSimpleMessage(js.getEmailId(),
25. if (ij.contains(Integer.parseInt(jobId)))
26. if (il.contains(Integer.parseInt(jobId)))
27. model.addAttribute("job", jp);
28. model.addAttribute("seeker", js);
29. model.addAttribute("company", company);
30. model.addAttribute("interested", i);
31. model.addAttribute("applied", j);

```

c. Method - cancelapplication

- i. Input : jobappId
- ii. Output : deletion confirmation
 1. Boolean deleted = return 0 or 1 after deletion
 2. if(deleted)
 3. Return deleted

d. Method - modifyApplicationState

- i. Input : jobappid, state
- ii. Output : modification confirmation
 1. Jobapplication ja = modify the application
 2. if(ja == null) "error"
 3. Else "modified"

e. Method - uploadstatus

1. Return "uploadstatus"

f. Method - getappliedjobs

- i. Input : id
- ii. Output : confirmation message
 1. Query query = createquery
 2. Query.setParameter - id
 3. List<Integer> list = new ArrayList<Integer>();
 4. List<> querylist = query.getResultList();
 5. for (Iterator<?> iterator = querylist.iterator(); iterator.hasNext();)
 6. int uid = (int) iterator.next();
 7. list.add(uid);
 8. Print list

g. Method - getAppliedjobs

- i. Input : jobseekerId
- ii. Output : list of applied jobs
 1. List<> jobSeekerAppliedList=getJobApplicationList();
 2. List<Integer> jobIdList = new ArrayList<Integer>();
 3. for (Iterator iterator = jobSeekerAppliedList.iterator(); iterator.hasNext();)
 4. JobApplication ja = (JobApplication) iterator.next();
 5. int jobId = ja.getJobPosting().getJobId();
 6. jobIdList.add(jobId);

5. Class Name - Jobpostingcontroller

a. Method - showHomePage

- i. Input : cid, model
- ii. Output : postjob
 1. Company company = get company details
 2. Set model according to the company id
 3. Set model of each company

b. Method - createJobPosting

- i. Input : cid, model, title, description, responsibilities, location, salary
- ii. Output : jobprofile, error
 1. JobPosting j = new JobPosting();
 2. j.setTitle(title);
 3. j.setDescription(description);
 4. j.setResponsibilities(responsibilities);
 5. j.setLocation(location);
 6. j.setSalary(salary);
 7. j.setKeywords(title + " " + description + " " + responsibilities + " " + location);
 8. try JobPosting p1 = create job posting based on the company id

9. Add job in the model
10. Company company = get company details
11. model.addAttribute("company", company);
12. return "jobprofile";
13. catch (Exception e)
14. HttpHeaders httpHeaders = new HttpHeaders();
15. Map<String, Object> message = new HashMap<String, Object>();
16. Map<String, Object> response = new HashMap<String, Object>();
17. message.put("code", "400");
18. message.put("msg", "another passenger with the phone number already exists.")

c. Method - deletejobposting

- i. Input : id, model
- ii. Output : confirmation message
 1. if (jobDao.deleteJobPosting(id))
 2. String message = "Job Posting with JobID " + id + " is deleted successfully";
 3. model.addAttribute("message", message);
 4. return "message";
 5. else
 6. return "error"

d. Method - showUpdatepage

- i. Input :id, cid, model
- ii. Output : updatejob
 1. Company company= select the company using the cid
 2. Get job potings using the id
 3. Add job to the model
 4. Add company to the model

e. Method - updateJobposting

- i. Input : cid, model, title, description, responsibilities, location, salary
- ii. Output: updated profile
 1. if (job != null)
 2. job.setjobId(id);
 3. job.setDescription(description);
 4. job.setState(Integer.parseInt(state));
 5. job.setTitle(title);
 6. job.setLocation(location);
 7. job.setResponsibilities(responsibilities);

8. JobPosting p1 = jobDao.updateJobPosting(job);
9. model.addAttribute("job", p1);
10. Company company = companyDao.getCompany(Integer.parseInt(cid));
11. model.addAttribute("company", company);

f. Method- modifyjobstate

- i. Input: jobid, state
- ii. Output: confirmation message
 1. Get job postings using the job id
 2. Set state of job
 3. Update job by calling updateJobposting function and sending the job id

6. Class name - Mailcontroller

a. Method - createMail

- i. Input : model
- ii. Output : confirmation of sent mail
 1. String action =
request.getRequestURL().substring(request.getRequestURL().lastIndexOf("/") + 1);
 2. Map<String, String> props = labels.get(action);
 3. Set<String> keys = props.keySet();
 4. Iterator<String> iterator = keys.iterator();
 5. while (iterator.hasNext()) {
 6. String key = iterator.next();
 7. model.addAttribute(key, props.get(key));
 8. model.addAttribute("mailObject", new MailObject());

b. Method - createMail

- i. Input: model, mailObject, errors
- ii. Output : return to home page
 1. if (errors.hasErrors())
 2. return "mail/send";
 3. emailService.sendSimpleMessage(mailObject.getTo(), mailObject.getSubject(),
mailObject.getText());

7. Class Name - MainController

a. Method - showHomepage

- i. Output : index page

b. Method - showRegisterpage

- i. Output : register page

c. Method - login

- i. Input : emailid, password, type, model
- ii. Output: profile of company or the seeker
 1. if (type.equals("recruiter"))
 2. list = companyDao.PasswordLookUp(email);
 3. if (list.size() == 0)
 4. model.addAttribute("message", message);
 5. return "index";
 6. else
 7. if (pwd.equals(list.get(0)))
 8. List<Integer> cidl = new ArrayList<Integer>();
 9. cidl = companyDao.getCompanyIdFromEmail(email);
 10. Company cmp = companyDao.getCompany(cidl.get(0));
 11. model.addAttribute("company", cmp);
 - 12.
 13. return "companyprofile";
 14. else if (type.equals("seeker"))
 15. list = jobSeekerDao.PasswordLookUp(email);
 16. if (list.size() == 0)
 17. model.addAttribute("message", message);
 18. return "index";
 19. else
 20. if (pwd.equals(list.get(0)))
 21. List<Integer> jsl = new ArrayList<Integer>();
 22. jsl = jobSeekerDao.getUserIdFromEmail(email);
 23. JobSeeker js = jobSeekerDao.getJobSeeker(jsl.get(0));
 24. model.addAttribute("seeker", js);
 25. return "userprofile";
 26. System.out.println(list);
 27. model.addAttribute("message", message);

d. Method - verification

- i. Input : type, pin, user id, model
- ii. Output :
 1. if (type.equals("seeker"))
 2. JobSeeker j = jobSeekerDao.getJobSeeker(userId);
 3. if (j.getVerificationCode() == pin)
 4. j.setVerified(true);
 5. jobSeekerDao.verify(j);

```

6.         model.addAttribute("seeker", j);
7.         return "userregister";
8.     else
9.         return "error";
10. else
11.     Company j = companyDao.getCompany(userId);
12.     if (j.getVerificationCode() == pin)
13.         j.setVerified(true);
14.         companyDao.verify(j);
15.         model.addAttribute("company", j);
16.         return "companyregister";
17.     else
18.         return "error";

```

8. Class name - EmailService

- a. **Method** - sendSimpleMessage
 - i. Input : to,subject,text
- b. **Method** - sendSimpleMessageUsingTemplate
 - i. Input : to,subject,template,templateArgs
- c. **Method** - sendMessageWithAttachment
 - i. Input : to,subject,text,pathToAttachment

9. Class name - EmailServiceImpl

- a. **Method** - sendSimpleMessage
 - i. Input : to,subject,text
 - ii. Output : send message
- b. **Method** - sendSimpleMessageUsingTemplate
 - i. Input : to,subject,template,templateArgs
 - ii. Output : sendSimpleMessage
- c. **Method** - sendMessageWithAttachment
 - i. Input : to,subject,text,pathToAttachment
 - ii. Output : send message

10. Class name - MailObject

- a. **Method** - getTo
 - i. Output : to
- b. **Method** - setTo
 - i. Input : To
- c. **Method** - getSubject

- i. Output : subject
- d. Method - setSubject**
 - i. Input : subject
- e. Method - getText**
 - i. Output : Text
- f. Method - setText**
 - i. Input : text

11. Class name - JobSeeker

- a. Method - getJobSeekerId**
 - i. Output : JobSeekerId
- b. Method - setJobSeekerId**
 - i. Input : JobSeekerId
- c. Method - getFirstName**
 - i. Output : firstName
- d. Method - setFirstName**
 - i. Input : firstName
- e. Method - getLastName**
 - i. Output : lastName
- f. Method - setLastName**
 - i. Input : lastName
- g. Method - getEmailId**
 - i. Output : emailId
- h. Method - setEmailId**
 - i. Input : emailId
- i. Method - getPassword**
 - i. Output :password
- j. Method - setPassword**
 - i. Input : password
- k. Method - getWorkEx**
 - i. Output : workEx
- l. Method - setWorkEx**
 - i. Input : workEx
- m. Method - getHighestEducation**
 - i. Output : highestEducation
- n. Method - setHighestEducation**
 - i. Input : highestEducation
- o. Method - getSkills**

i. Output : skills

p. Method - setSkills

i. Input : Skills

q. Method - isVerified

i. Output : verified

r. Method - getVerificationCode

i. Output : verificationCode

s. Method - setVerified

i. Input : verified

t. Method - setVerificationCode

i. Input : verificationCode

u. Method - getInterestedjobs

i. Output : interestedjobs

v. Method - setInterestedjobs

i. Input : interestedjobs

w. Method - getJobApplication

i. Output : jobApplicationList

x. Method - setJobApplication

i. Input : jobApplicationList

10. Interface - CompanyDao

A. Method - PasswordLookUp

@param emailid

@return password for the given emailid

B. Method - Company createCompany

@param com

@return Created company

@throws Exception

C. Method - Company updateCompany

@param js

@return Updated company

D. Method - Company getCompany

@param id

@return Company

E. Method - void verify

@param c

F. Method - List<?> getJobsByCompany

@param companyId

@param state

@return List of jobs according to the state

G. Method - List<Integer> getCompanyIdFromEmail

@param emailid

@return

11. Interface - InterestedDao

A. Method - Interested createInterest

@param in

@return Created interest

@throws Exception

B. Method - boolean deleteInterest

@param id

@return true if interest has been deleted

C. Method - Interested getInterest

@param id

@return Interest

D. Method - List<?> getInterestedJobId

@param jobId

@param userId

@return List of the job ids of the jobs the user is interested in

E. Method - List<Integer> getAllInterestedJobId

@param userId

@return List of the job ids of the jobs the user is interested in

12. Interface InterviewDao

A. Method - Interview createInterview

@param jobseekerid

@param company

@param location

@param datetime

@param flag

B. Method - String acceptInterview

@param jobseekerid

C. Method - List<interview> getAllInterviews

@param jobseekerid

13. Interface JobApplicationDao

A. Method - JobApplication apply

@param jobseekerId

@param jobId

@param resumeFlag

@param resumePath

@return The newly created job application

B. Method - JobApplication getJobApplication

@param jobAppld

@return Required job application

C. Method - boolean cancel

@param jobAppld

@return True if the application was successfully cancelled

D. Method - JobApplication modifyJobApplicationStatus

@param state

@return Modified job application

E. Method - JobApplication updateApplication

@param ja

@return Updated job application

14. Interface JobPostingDao

A. Method - JopPosting createJobPosting

@param job
@param cid
@return New JobPosting
@throws Exception

B. Method - JobPosting getJobPosting

@param id
@return Requested JobPosting

C. Method - boolean deleteJobPosting

@param id
@return True if JobPosting is Deleted

D. Method - JobPosting updateJobPosting

@param job
@return Updated Job Posting

15. Interface JobSeekerDao

A. Method - List<?> filterJobs

-> @param jpv
-> @param joblds
-> @return Job Postings according to the provided parameter

B. Method - JobSeeker createJobSeeker

-> @param job
-> @return new job seeker
-> @throws Exception

C. Method - JobSeeker updateJobSeeker

-> @param js
-> @return updated job seeker

D. Method - List<String> PasswordLookUp

-> @param emailid
-> @return password

E. Method - void verify

->@param j

F. Method - List<?> searchJobs

->@param searchString

->@return Jobs for that search string

G. Method -List<Integer> getUserIdFromEmail

->@param emailid

->@return userId

16. Class - CompanyDaoImpl implements CompanyDao

-> Private Variables EntityManager entityManager using annotation - PersistenceContext

-> @Override - annotation

Method - public List<String> PasswordLookUp

->@param emailid

-> **Database query** ="SELECT password FROM Company c WHERE c.companyUser = :emailId "

-> **Set parameter to be passed as "EmailId"** -> query.setParameter("emailId", emailid);

-> new ArrayList<String>()

-> query.getResultList()

for (**iterate over the query.list**)

-> **to add password to the list** -> list.add(pwd)

-> print("list :::::::::::::::::::: " + list)

-> **@return list**

-> @Override

Method - public List<Integer> getCompanyIdFromEmail

->@param emailid

->**Database query** ="SELECT companyId FROM Company c WHERE c.companyUser = :emailId "

->**Set parameter to be passed as** -> emailid

->new ArrayList<Integer>();

->query.getResultList();

for (**iterate over the queryList**)

-> **To add Company Id to the list** -> cid

-> **return list**

-> @Override

//this method handles all the exception that could be thrown

Method - public Company createCompany

- > @param company c
- > throws exception
- > try -> entityManager.persist(c);
- > catch -> Exception -> e.printStackTrace()
- > **return c**

-> @Override

Method public Company getCompany

- > @param id
- > Company js -> null
- > js -> entityManager.find(Company.class, id)
- > **return js**

-> @Override

Method public Company updateCompany(Company js) {

- > Company c -> getCompany -> @param js.getId()
- > c.setCompanyName -> @param js.getCompanyName()
- > c.setCompanyUser -> @param js.getCompanyUser()
- > c.setDescription -> @param js.getDescription()
- > c.setHeadquarters -> js.getHeadquarters()
- > c.setVerified -> js.isVerified()
- > try -> if (c is not null)
 - Then** -> entityManager.merge(c)
- > catch -> Exception e -> e.printStackTrace()
- > **return c**

-> @Override

Method - public void verify(Company c) {

- > Company c1 -> getCompany -> @param c.getId()
- > c1.setVerified -> @param -> c.isVerified();
- > try -> if (c not equal to null) Then -> entityManager.merge(c1)
- > catch -> Exception e -> e.printStackTrace()

-> @Override

Method -> public List<?> getJobsByCompany -> @param company

-> **Database query to be passed** -> SELECT jobId, title, description, responsibilities, location, salary, state, companyId, companyName FROM JobPostingsView jp WHERE jp.companyId = :companyId

-> **Parameters to be sent in query** -> companyId

-> query.getResultList()

-> **return querylist**

17. Class - public class InterviewDaoImp -> implements InterviewDao {

Variables -> private -> EntityManager entityManager

-> @Override

Method - public Interview createInterview()

-> @param jobseekerid

-> @param company

-> @param location

-> @param datetime

-> @param flag

-> new Interview()

-> interview.setCompany() -> @param company

-> interview.setJobseekerid() -> @param jobseekerid

-> interview.setDatetime() -> @param datetime

-> interview.setLocation() -> @param location

-> interview.setFlag() -> @param string "false"

-> entityManager.merge() -> interview

-> return interview

-> @Override

-> **Method** public String acceptInterview()

-> @param jobseekerid

-> new Interview()

-> interview.setFlag() -> @param string as "true"

-> **Database Query updation** -> UPDATE interview SET flag = true WHERE jobseekerid= :id

-> query.setParameter() -> @param "id", jobseekerid

-> entityManager.merge() -> interview

-> **return "updated"**


```

-> @Override
-> Method - public List<Interview> getAllInterviews()
        -> @param jobId -> int
        -> Database query -> SELECT company, location, time FROM interview WHERE jobId = :jobId
        -> return null;

```

18. Class public class JobPostingDaoImpl -> implements JobPostingDao

```

-> @PersistenceContext -> private EntityManager entityManager;
-> @Override
-> Method -> public JobPosting createJobPosting(JobPosting job, int cid) {
        -> try -> System.out.println("1");
        -> Company c -> entityManager.find() -> @param Company.class, cid
        -> job.setCompany(c);
        -> System.out.println("2");
        -> entityManager.persist(job);
        -> System.out.println("3");
        -> catch -> Exception e -> e.printStackTrace()
        -> return job

```

```

-> @Override
-> Method -> public JobPosting getJobPosting()
        @Param id
        -> JobPosting j -> null
        -> j -> entityManager.find() -> @param JobPosting.class, id
        -> return j

```

```

-> @Override
-> public boolean deleteJobPosting(int id) {
        -> JobPosting p -> getJobPosting() -> @param id
        -> if (p is not equal to null) -> entityManager.remove() -> @param p
        -> else -> return -> false
        -> return true

```

```

-> @Override

```

```
-> Method public JobPosting updateJobPosting()  
    -> @param -> JobPosting p  
    -> JobPosting p1 -> getJobPosting() -> @Param p.getJobId  
    -> p1.setDescription() -> @param p.getDescription()  
    -> p1.setLocation() -> @param p.getLocation()  
    -> p1.setResponsibilities() -> @param p.getResponsibilities()  
    -> p1.setSalary() -> @param p.getSalary()  
    -> p1.setState() -> @param p.getState()  
    -> p1.setTitle() -> @param p.getTitle()  
    -> try -> if (p1 != null) -> entityManager.merge() -> @param p1  
    -> catch -> Exception e -> e.printStackTrace()  
    -> return p1
```