Final Report – Company Scheduler

Team Satisfaction: Kyle V, Patrick K, Raamish S, Nikhil D.

1. Implementation Report

Requirement	***Implementation Plan***	***Progress***	***Comments***
IN001	Create a login for both am employee and employer	100%	
IN002	Retrieve the schedule of an employee	100%	
IN003	Make changes to an employee's schedule	100%	
IN004	An employee should be able to input/change their availability	100%	
IN005	An admin account (employer) should be able to add an employee into the database	100%	
IN006	An employee can request days/time off	100%	
IN007	An employer can accept/deny requests for time off	100%	
IN008	The employer should be able to generate a new weekly schedule	100%	
IN009	An employer can set the max number of hours that an employee can work	50% (Reworked this idea as explained in comments)	Changed this from our initial plans; instead of the employer setting the max number of hours, the max number of hours is restricted by the attribute (employee-type) which is set by

			the employer. (Full time max = 40, part-time <40)
IN010	An employer can remove an employee from the system	100%	
IN011	An employer can make changes to an employee's profile	100%	
IN012	Logout works successfully for both employees and employers	100%	
IN101	Downtime should not exceed more than one minute in a day	100%	
IN102	Runs on both MacOS and Windows OS as a desktop application	75%	Mostly works as intended, but sometimes the updated GUI has bugs on MacOS. Does not get in the way of functionality, though.
IN103	Any interaction between the user and system should not take more than 10 seconds to register in the database	100%	
IN104	Only employers can view information of all their employees (Employees may not look at each other's profiles)	100%	

2. Building and Running Instruction

Building our program is very simple. It is a desktop application. Go to our Github Repository, download “Company Scheduler.jar” and run that file.

Alternatively, you can download our repository, navigate to and compile Main.java and run it, but running the .jar file is much simpler and easier.

To log into our admin account, our username is “admin” and our password is “admin”. From here, you can create an employee account through the add employee feature, and use the login you created for that employee to log into the employee side of the client. If you would not like to add another employee into the database, we have provided an employee account login below:

Username: kupcha

Password: password123

3. Bug Report

Non-functional requirements bugs:

- Some of the UI fonts do not transfer well to Mac OS (Can make it buggy to run on Mac OS)

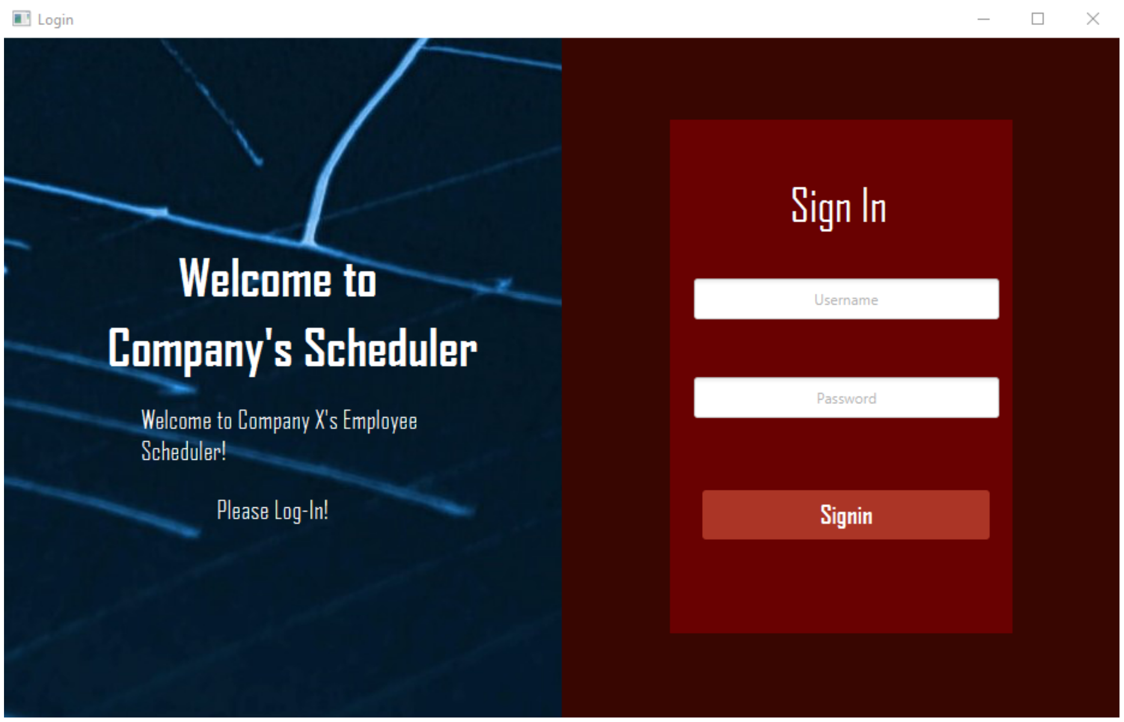
Functional Requirement Bugs:

- Scheduling conflicts can occur within the program if there are not enough available employees, as the algorithm takes only employee-type and availability into account.
- For requesting time off, inputting an incorrectly formatted String will cause an error.
- When changing availability to “open”, all other boxes should be automatically unchecked, but sometimes this does not occur, leaving the user to manually uncheck the rest of the boxes.
- When an employee requests off and gets to the week of, there’s an issue with scheduling other employees on the day.

4. Screen Shot

Login screen:

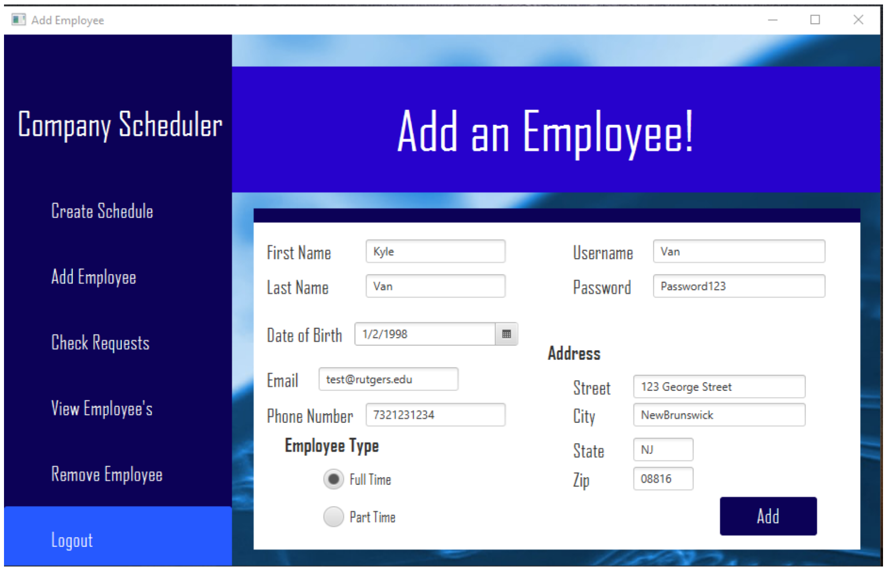
This is the main login screen for our application. At this screen, a user can login as either an employer or an employee. Must click sign-in to sign in.



Employer (admin) logged in for the below –

Add Employee:

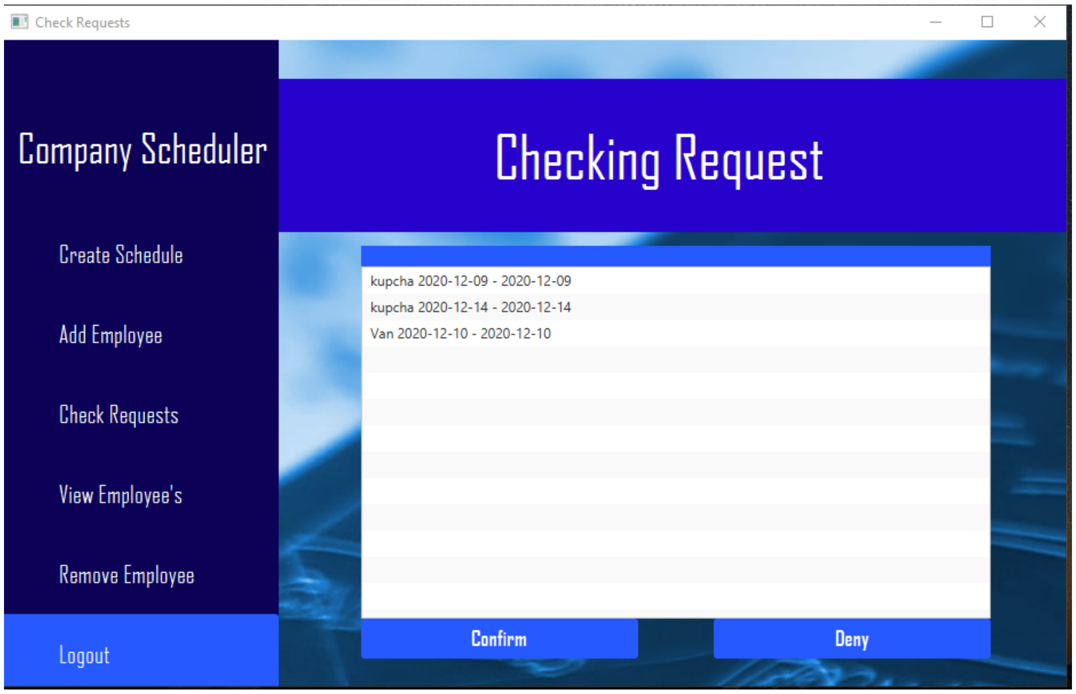
This is the add employee feature. Here, an employer can add an employee and assign values to all these attributes for them as well.



First Name: Kyle
Last Name: Van
Date of Birth: 1/2/1998
Email: test@rutgers.edu
Phone Number: 7321231234
Employee Type: [✓ Full Time] Part Time
Username: Van
Password: Password123
Address
Street: 123 George Street
City: NewBrunswick
State: NJ
Zip: 08816

Check Requests:

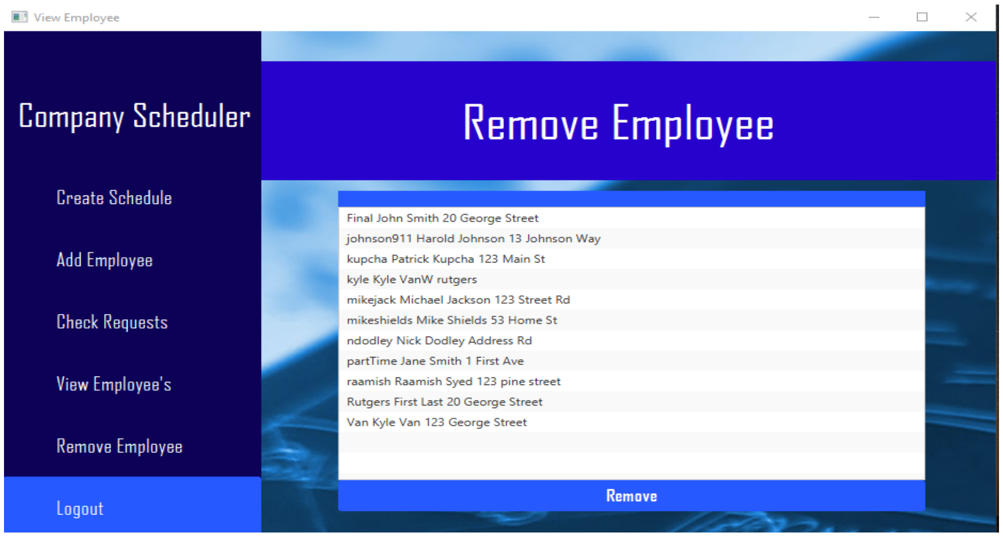
An employer can check and then accept/deny requests for time off from employees. On this screen, it lists ALL requests that employees have made for time off.



kupcha 2020-12-09 - 2020-12-09
kupcha 2020-12-14 - 2020-12-14
Van 2020-12-10 - 2020-12-10

Remove Employee:

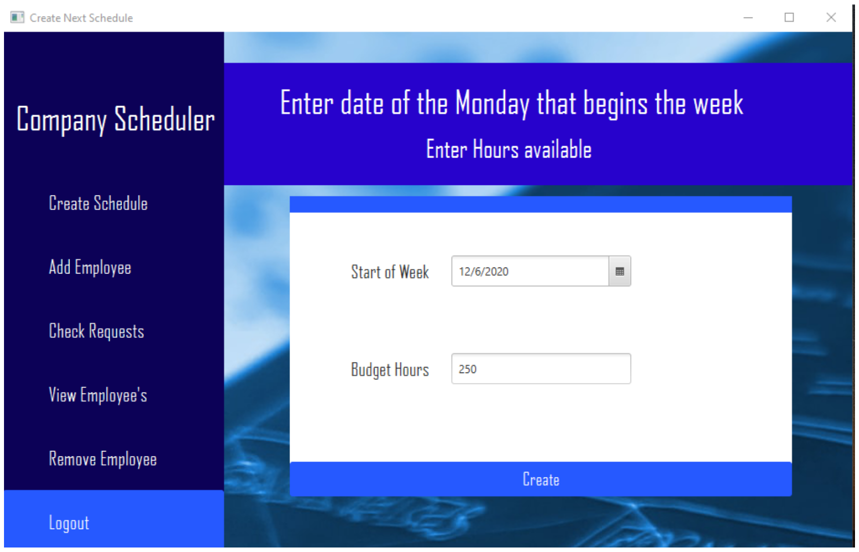
An employer can remove an employee from the system. This very simply removes the employee from the database and erases their login.



Final John Smith 20 George Street
johnson911 Harold Johnson 13 Johnson Way
kupcha Patrick Kupcha 123 Main St
kyle Kyle VanW rutgers
mikejack Michael Jackson 123 Street Rd
mikeshields Mike Shields 53 Home St
ndodley Nick Dodley Address Rd
partTime Jane Smith 1 First Ave
raamish Raamish Syed 123 pine street
Rutgers First Last 20 George Street
Van Kyle Van 123 George Street

Create Schedule:

An employer can generate a new weekly schedule inputting a start date and budget hours for the week.



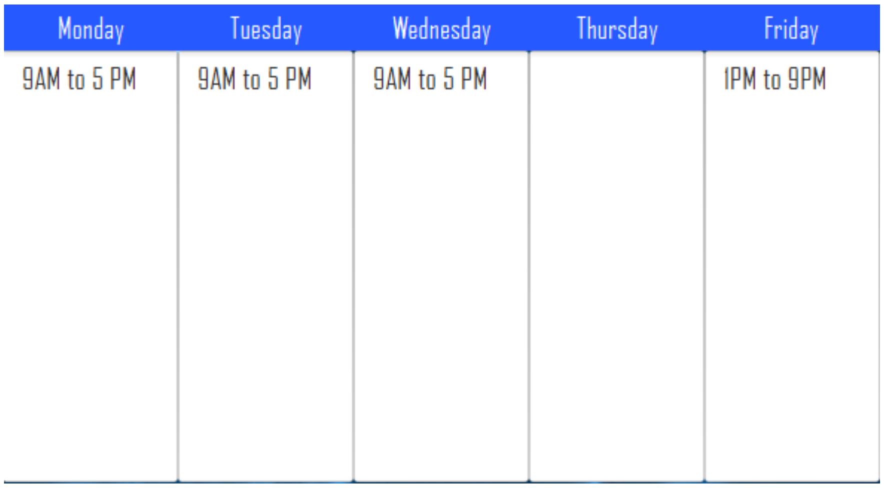
Start of Week 12/6/2020

Budget Hours 250

Employee logged in for the below –

View Schedule:

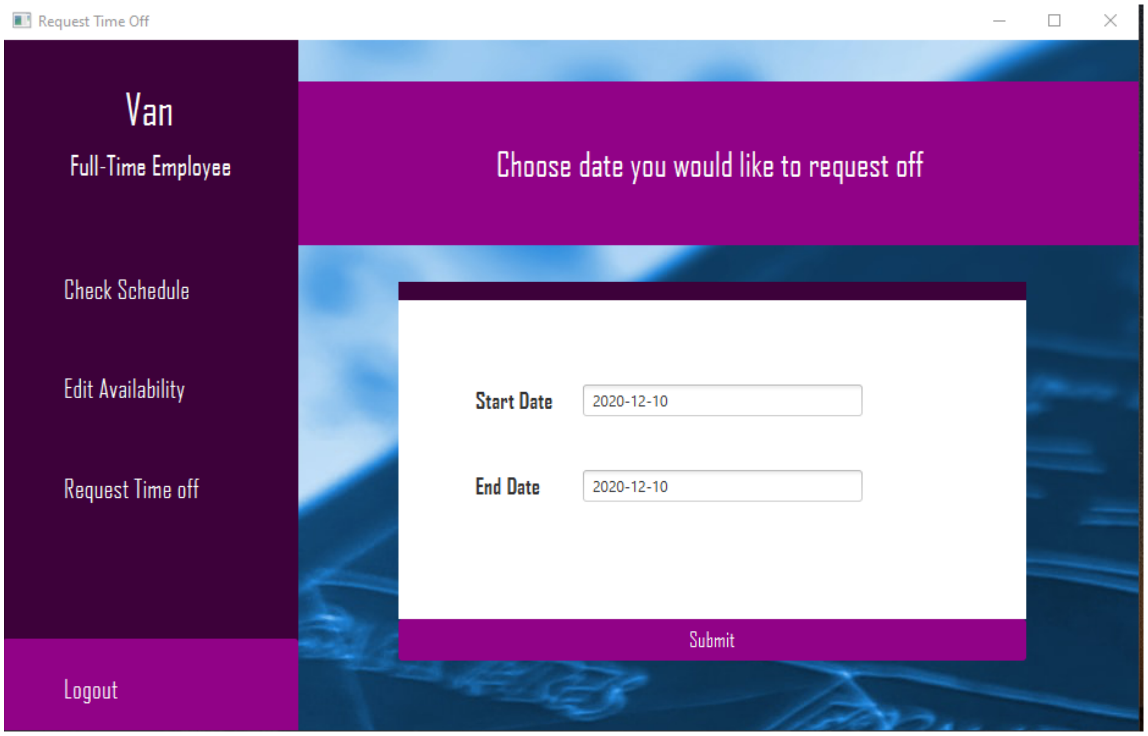
An employee can view their schedule for the week and see what shifts they have to work.



Monday	Tuesday	Wednesday	Thursday	Friday
9AM to 5 PM	9AM to 5 PM	9AM to 5 PM		1PM to 9PM

Request Time Off:

An employee can request time off and it will send this request to the employer-side of the client to be either accepted or denied by an employer.



Start Date 2020-12-10

End Date 2020-12-10

5. Project Metrics

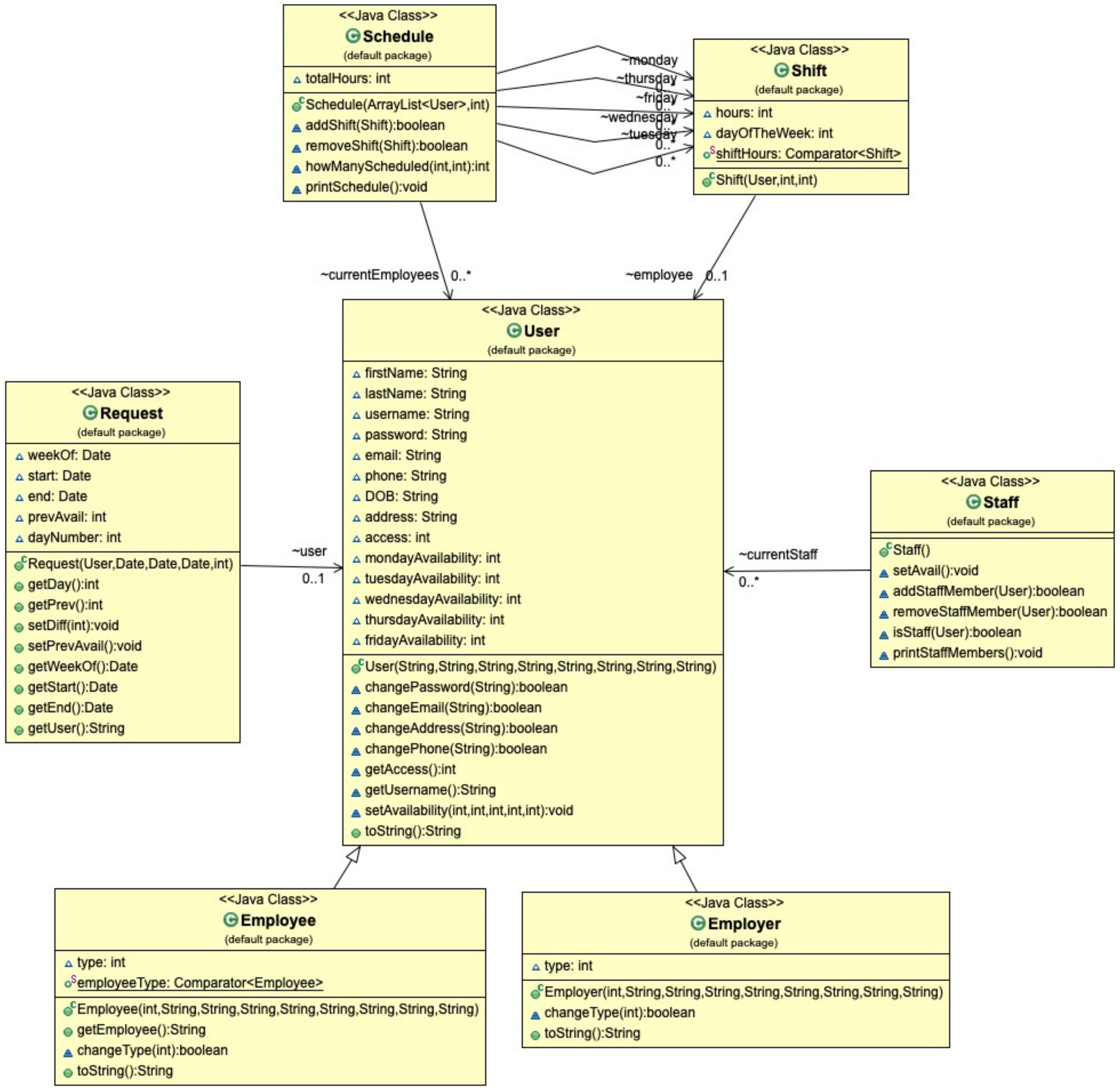
- Main.java
- Entity Classes: 7
- FXML Controllers: 12
- Database Controller

- Lines of code (Java): 4,698
- Lines of code (FXML): 1,993
- 6691 total lines of code
- 167 Comments .java
- 800 Blank Lines .java
- Entity Classes Methods: 42
- Controller Class Methods: 117

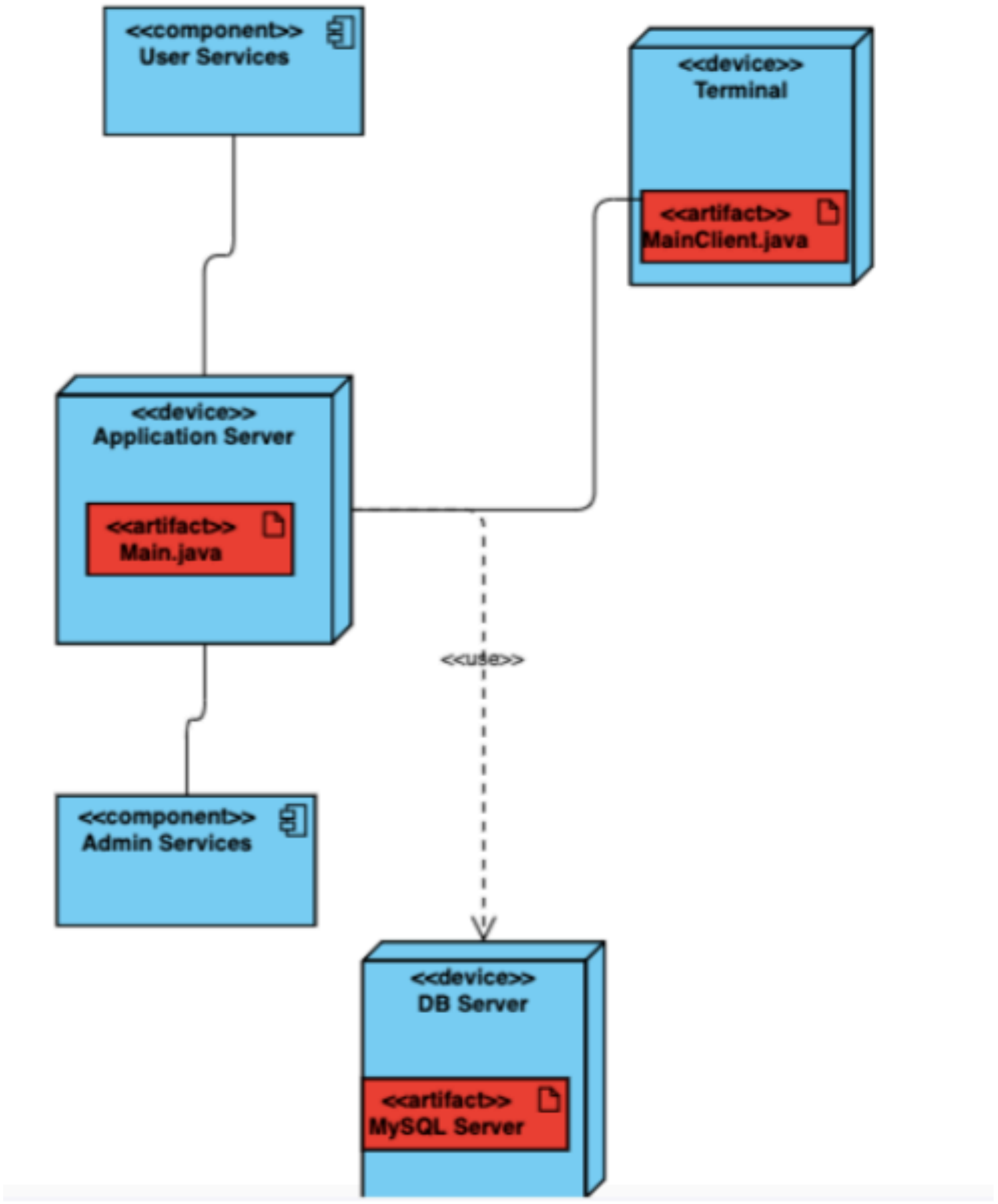
- java.util.*
- java.time.*
- java.sql.*
- javafx.*
- Auto Generated: 1642
- Manual: 2089

6. Software Structure and System Architecture

Modified Class Diagram:



Modified Deployment Diagram:



7. Summary

We accomplished the majority of our initial plans, while tweaking/changing a few as we progressed through the sprints. Being adaptive is definitely an important skill to develop in this type of work-environment. Initially, we wanted to make a scheduling application that uses a scheduling algorithm we created in order to automate employee scheduling for employers. There would be two login types to our application, an admin (employer) and user (employee). This was included in our final program/release. In our initial plans, we also wanted to have employees able to request time off and for employers to see and then accept or deny those requests. We also included this feature in our final release. As you can see from our requirements table in Section I, we finished all the SRS requirements we listed out, with the exception of a few. One major thing we changed was our GUI. Our GUI was far too simple when we started out, so we made it more complex and visually appealing in order to make our product look nicer overall. We also let the system decide the max number of hours an employee can work based on their employee-type, rather than an employer having to manually put in the maximum number of hours. This was nice quality of life update. Lastly, I think the largest thing we changed from our initial plans is that employees now enter their own availability. Initially, we had employers adding each employee's availability for them. However, this change makes it easier for the employer to add an employee, and it just made more sense to us to have each employee enter their own availability. Something we wanted to add but didn't have enough time to was to allow employees to switch shifts with each other within the system. If we were to keep developing this project, this would be a feature we would look to add in the future.

Contributions/Self-Evaluation:

- Manually scheduling can be a headache
- Automating employee schedules save a lot of time for employers
- There will be no errors in the scheduling because the algorithm takes care of it, thus issues regarding miscommunication between employers and employees of when their shifts are will be mitigated
- In turn, this would result in happier employees which would result in better reputation for the company
- Overall, it saves employers and HR departments A LOT of time and the only thing they have to worry about is manually checking requests for time off, which doesn't take too long.
- Evaluating our software; there are bugs of course, but the main functionality of the software is not compromised.
- These bugs could be fixed and the program could definitely be improved upon, given more time

- Our project could be a very useful asset to small businesses / growing companies!

Lessons Learned	Mistakes to Avoid
One major thing we learned from this whole class was how to work in an agile work environment. We learned how to use Jira + Confluence in a collaboration with our groups to keep ourselves on track. These tools/software are definitely very useful and popular in the software engineering field!	Make sure to COMMENT CODE as you're coding. It makes it a lot easier to go back and find things you're looking for. It also makes your code more organized and easier to read for someone else.
Learned how to use SceneBuilder to make more complex and interesting GUIs	Distribute the workload evenly throughout a Sprint! Doing all the coding a few days before the sprint ends is not optimal. Time management is key.
Learned a lot about using MySQL and JDBC and how to host/use a database created through AWS	When assigning a user story, it's better to overestimate the points (how long it will take) rather than underestimate
Learned a lot about how to properly and efficiently test our software. We also learned many different methods of testing.	
Overall, we gained a better understanding of how software engineering actually looks like in work-environment.	