Sprints

1. Start-Up

Sprint backlog:

1. Sketch the design of the final product
2. Design the main page using ASP.NET MVC
3. Design the navigation bar using ASP.NET MVC
4. Create a class for writing and reading from the database using JDBC to connect the second tier and the third tier.
5. Connect ASP.NET project to a JAX-WS to test the web service set-up.

Results and major changes/difficulties:

We encountered difficulties in connecting asp.net to jax-ws so we were stuck in this sprint for longer than expected

2. Sign Up and Log in

Included user stories:

1. As a user, I want to be able to sign up and log in so I can have my own account in the system.

Sprint backlog:

1. Create a class diagram containing classes needed for this sprint.
2. Create database table: User.
3. Create a model class: User.
4. Write a DAO in java to write a user object to the database.
5. Write the business logic required to check the user info is valid in Java.
6. Add web methods to the web service to connect the two tiers.
7. Throw customized exceptions when the sign-up form contains invalid data.
8. Create the user interface (Add sign-up option in the navigation bar and create the sign-up form and add action listeners to post the form to the web service) using ASP.NET MVC.
9. Write a DAO in java to read a from the user table in the database and check if the login credentials exist.
10. Throw customized exceptions when the login credentials contain invalid data such as the user not found in the database.

11. Create the user interface (Add login option in the navigation bar and create the login form and add action listeners to post the form to the web service) using ASP.NET MVC.

End of sprint reevaluation:

At the end of the sprint, we tried running the different tiers on separate computers, seeing as this was successful, we could now work on separate on the different tiers from different computers.

3. Create Service Offer and Requests

Included user stories:

1. As an employer, I want to be able to post job offers, so the workers would see it and ask to be hired.

2. As a worker, I want to be able to post job requests, so the employers would see it and hire me.

Sprint backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Create tables to store the service offers and requests in the database.
3. Create model classes: Post, ServiceOffer, ServiceRequest
4. Write a DAO to write service offer objects to the database.
5. Write the business logic required to check if the service offer is valid before storing it.
6. Throw customized exceptions when the service offer contains invalid data.
7. Add web methods to the web service to connect the two tiers.
8. Create the user interface (Add 'create service offer' option in the navigation bar and create the 'create service offer' form and add action listeners to post the form to the web service) using ASP.NET MVC.
9. Write a DAO to write service request objects to the database.
10. Write the business logic required to check if the service request is valid before storing it.
11. Throw customized exceptions when the service request contains invalid data.
12. Create the user interface (Add 'create service request' option in the navigation bar and create the 'create service request' form and add action listeners to post the form to the web service) using ASP.NET MVC.

End of sprint reevaluation:

At the end of this sprint, we decided to work on the View Service Offers and Requests so the user can immediately see the list once they create a service

4. View Service Offers and Service Requests

Included user stories:

1. As a user, I want to be able to see nearby job offers and request, so I can be able to apply for it or contact who posted it.

Sprint Backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Write a DAO to read service offer and requests objects from the database that are not expired yet and in the same county as the user.
3. Add a web method to the web service to connect the two tiers and return the service offers and services requests retrieved from the DB to the 1st tier.
4. Create the user interface (Add 'View Service Offers' and 'View Service Requests' option in the navigation bar and create the 'View Service Offers' page which shows the offers in a list) using ASP.NET MVC.

End of sprint reevaluation:

At the end of this sprint, we thought it'd be good for the logged-in user to see a list containing only what service requests/offers they created, so we reprioritized the user stories, to make sure to implement that before the next sprint (Apply and unapply for a service offer).

5. View *Logged-In* user’s Service Offers and Requests

Included user stories:

1. As a user, I want to be able to manage my service offers, to make sure that I can delete them when I no longer offer that service.

2. As a user, I want to be able to manage my service requests, to make sure that I can delete them when I no longer request that service.

Sprint Backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Write a DAO to read service offers objects that belong to the user from the database.
3. Add web methods to the web service to connect the two tiers.
4. Create the user interface (Add ‘View My Service Offers’ option in the navigation bar and create the ‘View My Service Offers’ page which shows the offers in a list) using ASP.NET MVC.
5. Write a DAO to read service requests objects that belong to the user from the database.
6. Create the user interface (Add ‘View My Service Requests’ option in the navigation bar and create the ‘View My Service Requests’ page which shows the requests in a list) using ASP.NET MVC.

End of sprint reevaluation:

1. At this point, the next sprint was missing a page where the user can see what they have applied for, so we added a new user story so the user can manage their applications.
2. Also, the user should be able to delete their posts, therefore we added some user stories that describe that, and we decided to work on that after finished the sprint of managing profiles.

6. Apply and unapply for a Service Offer

Included user stories:

1. As a worker, I want to be able to apply for job offers, so the employer that I am interested in the job offer and hire me.

2. As a worker, I want to view the service offers I applied for, so I can keep track of what I have applied for.

3. As a worker, I want to unapply to service offers, so if I changed my mind about my application, I can just unapply.

Product Backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Write model classes needed to complete this sprint.
3. Create a database table to store applicant description.
4. Write a DAO in java to write and delete an applicant object from the database.
5. Add web methods to the web service to connect the two tiers.
6. Add an 'Apply' link next to each of the service offers in the 'View Service offers' page
7. The 'Apply' link takes the user to another page where they can view the details and apply for the service offered using ASP.NET MVC.
8. Create a page with a list of the services offers that the user has applied for.
9. Add an unapply link next to each item of the services offers in the page mentioned last.

End of sprint reevaluation:
Continue as planned with the next sprint

7. View and Edit the *Logged-In* user's Profile and Add a description to it.

Included user stories:

1. As a user, I want to be able to handle my account/profile.

    a. As a user, I want to be able to edit my profile, to make sure that it's always up to date.

    b. As a user, I want to be able to edit my account information, to make sure that it's always up to date.

Sprint Backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Write model classes needed to complete this sprint.
3. Create a database table to store profile description.
4. Write a DAO in java to read and update a user object from the database.
5. Write a DAO in java to read, update a profile object from the database.
6. Add a web method to the web service to connect the two tiers and retrieve the user's profile.
7. Add a web method to the web service to connect the two tiers and submit user's new profile info.
8. Create the user interface (Add 'View/Edit My Profile' option in the navigation bar and create the 'View/Edit My Profile' page to view the retrieved data) using ASP.NET MVC.

End of sprint reevaluation:
Continue as planned with the next sprint

8. Delete logged-In user's service offer and request post

Included user stories:

1. As a user, I want to be able to manage my service requests, to make sure that I can delete them when I no longer request that service.

2. As a user, I want to be able to manage my service offers, to make sure that I can delete them when I no longer offer that service.

Product Backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Write DAO in java to delete service offer and request from database.
3. Add web methods to the web service to connect the two tiers.
4. Add a link next to the 'My Service Offers' list items to delete the offer.
5. Add a link next to the 'My Service Requests' list items to delete the request.

End of sprint reevaluation:
Continue as planned with the next sprint

9. View list of applicants to service offer post

Included user stories:

1. As an employer, I want to be able to view the list of applicants of my job offers, to know if anyone is interested in the service I'm offering.
2. As an employer, I want to be able to view the applicants' profiles, in order to decide who works better for the service offer.

Product Backlog:

1. Add the classes needed for this sprint to the class diagram.
2. Write DAO in java to read the applicants of a service offer.
3. Add a web method to the web service to connect the two tiers.
4. Add a link next to the 'My Service Offers' list items to 'view the applicants'.
5. Create a new page that the 'view the applicant' link to.

End of sprint reevaluation:
Continue as planned with the next sprint

10. Contact users :

Included User Stories

1. As an employer, I want to be able to contact the applicants of the job offer, so I can find out if I should pick any/some of them (so I can discuss the job details).

   Sprint backlog:

   1. Add the classes needed for this sprint to the class diagram.
   2. Write model classes needed to complete this sprint.
   3. Create a database table to store messages.
   4. Write a DAO in java to read and write messages to the database.
   5. Add a web method to the web service to connect the two tiers and read and write messages.
   6. Figure out a way to let the web service invoke methods on the first tier in order to let the users know when they receive messages.
   7. Create a page that contains the contact list
   8. Create a page for chatting.

   End of sprint Evaluation:

   At this point, we decided to stop implementing since this sprint couldn't be finished, and it was time to start writing the reports.

11. Employ User:

Included user stories:

1. As a user, I want to be able to pay/receive the payment to workers/ from employers securely, so I make sure I don't get scammed.

   a. As a worker, I want the employer to pay the payment as a deposit to the system, to make sure I will get paid after I finish the job.

   b. As a user, I want to acknowledge the completion of the job to the system, so the payment can proceed by the system paying the deposit, that the employer paid earlier, to the worker.

   c. As a user, I want to be able to report that the job didn't happen, so the payment doesn't happen, and the employer gets their deposit back.

   d. As a user, I want the administrator to solve payment issues in case there was mismatching between what the worker and employer reported after the job is supposed to be done, so the worker gets paid if they actually worked or so the employer get their money back in case the worker didn't really work.