**SW Engineering CSC 648/848**

**Shyft**

Service Industry Contract Work

**Section 4, Team 05**

Courtney Radford: Team Lead

Ana Navarro: Scrum Manager

Mohammad Khan: Github Manager

James Giatpaiboon: Front End Lead

Andy Ouyang: Back End Lead/M2 Editor

Jagjot Saggar: Database Manager

**“Milestone 2”**

October 22nd, 2021

Document Revision History Table

| ID # | DOCUMENT & Version/Release/Build # | DATE | DESCRIPTION | TRACKING NOTES |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

I. Data Definitions V2

Revision from Milestone 1

| **Name** | **Definition & Examples** | **Usage** |
| --- | --- | --- |
| Resume | a user’s resume that is uploaded for business owners to view | A list of these will be shown on a registered business owner with tags selected |
| Skills | categories that associate with jobs the users skilled at, e.g., *Management, Marketing, Java, Full Stack* | Registered users select skills as they register their account and registered business owners use skills to filter out specific ones they are looking for |
| Job | attributes that include job title, skills, salary, location, and type  *e.g, job title: bartender, skills: good communication, mixology, salary: $20 an hour, location: 1234 temp street SF, CA 94116, type: in-person* | Jobs will be posted by employers and contractors will be able to view these jobs and submit their applications for employers to view. |
| Shift | attributes include job title, start time, end time, pay rate ($ per hour), location, boolean item representing availability | Jobs will have shifts that represent a time period an employer wants an employee to work |
| User: Contractor | A person who signed up with the application, e.g., *Ben Li, age: 26, gender: male, skills: Databases, Java, Python* | Information from the user is collected and stored. The users information will be made available for business owners to view |
| User: Employer | A person who signed up with the application looking to hire new people, e.g, *Rocky Wade, age: 34, gender: male, business name: Foodverse* | Information from the business owner that is collected and stored. Admins can view this information to verify the business before allowing the business owner to view resumes |
| Account | An account with set permissions on whether it is a contractor or employer. | Users use an account to upload resumes and interact with business owners.  Business owners are able to view these resumes and contact the users |

Data Definitions continued

| **Primary Data Name** | **Sub Data** |
| --- | --- |
| Resume | * resume\_id * user\_id * resume\_path |
| Skills | n/a |
| Job | * job\_id * title * skills |
| Shift | * shift\_id * shift\_title * start\_time * end\_time * pay\_rate * location * availability |
| User: Contractor | * user\_id * name * username * password * email * usertype |
| User: Employer | * user\_id * name * username * password * email * usertype |
| Account | * username * password * email |

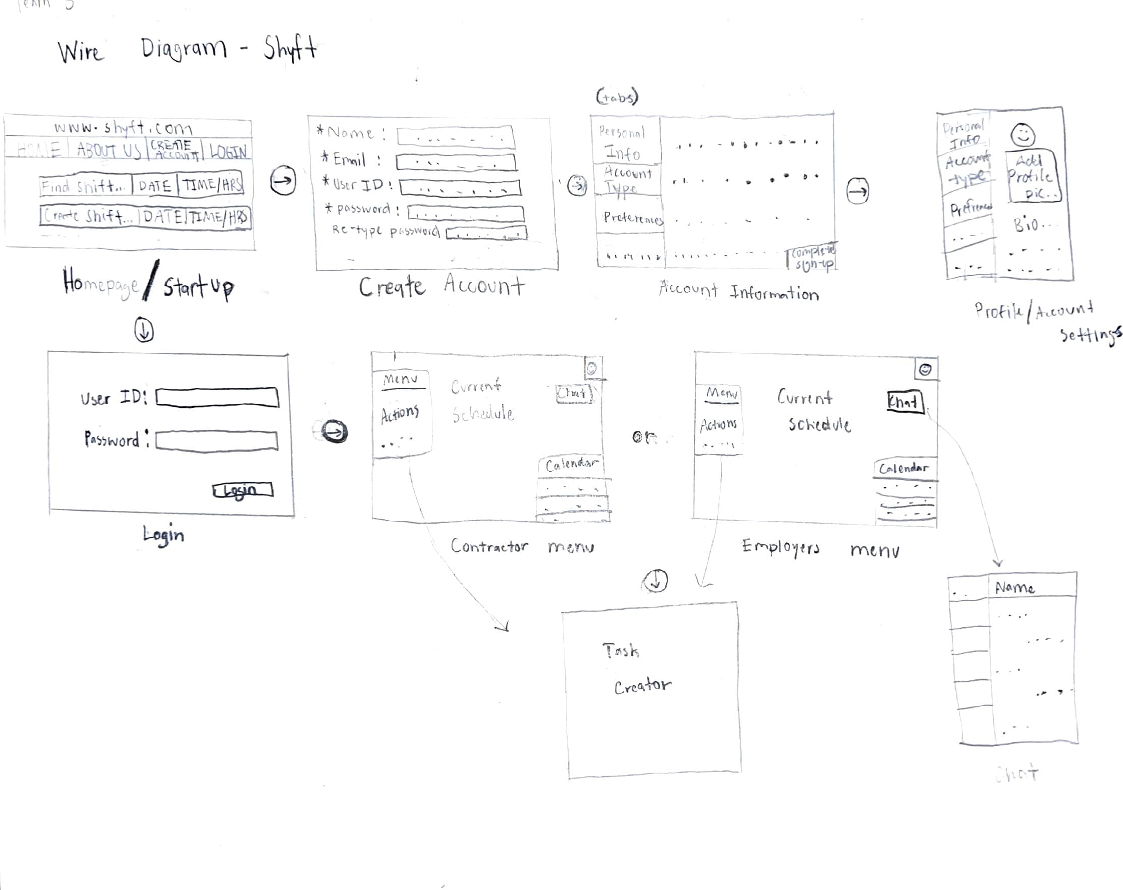
II. Functional Requirements V2

|  | Must-have |  | Desired |  | Opportunistic |
| --- | --- | --- | --- | --- | --- |

|  | ID | Functional Requirement Description | Details (As Needed) |
| --- | --- | --- | --- |
|  | 1 | Users can create and login to their accounts using a unique user id and password. | 1.1) user can sign up for a new account  1.2) user can sign in with their existing account |
|  | 2 | User can logout of their account. | 2.1) user can safely logout |
|  | 3 | Guest users, those who have not been verified, will only see gig listing previews. They will not be able to see full details or access our live scheduling system. |  |
|  | 4 | User will have a profile (one of two categories: contractor, employer). | 4.1) users will have the option to create an account for contractors or employers |
|  | 5 | User can edit and update their profile | 5.1) user can change their personal description  5.2) user can change their profile photo and cover photo  5.3) user can change their account’s password |
|  | 6 | Contractors can pick up, and drop shifts | 6.1) contractors can pick up shifts that will be taken down once a user has confirmed that they will be available for it  6.2) contractors can drop shifts that no longer work for them |
|  | 7 | Employers can create, open and close shifts | 7.1) employers can create shifts that contractors can pick up  7.2) employers can open shifts that need to be picked up  7.3) employers can close shifts that are no longer available so that they can be taken down from Shyft |
|  | 8 | Contractors can get paid through the app | 8.1) The app should be able to pay the Contractors once they finish their job |
|  | 9 | Users can view and select shifts through calendar view | 9.1) Contractor can select a shift through the calender view |
|  | 10 | Users can communicate through direct messages | 10.1) Contractor can communicate with employer through direct messages  10.2) Contractors can communicate with other contractors  10.3) Employer can communicate with other employers through direct messages |
|  | 11 | Users can get a recommendation about shifts or contractors available near the user | 11.1) Contractors get recommendations when logged in about shifts near them  11.2) Employees can get recommendations when logged in about Contractors near them |
|  | 8 | Contractors can trade their shifts with another contractor | 8.1) contractors can trade shifts with another user |
|  | 9 | Users can view and select shifts through calendar view | 9.1) users can view a calendar with shifts that are available on certain days with details about them  9.2) users can pick up shifts that will be available in the future |
|  | 10 | Users can communicate with each other through direct messaging | 10.1) users can send messages to other users  10.2) users can reply to messages that they receive from other users |
|  | 11 | Users can get a recommendation about shifts or contractors available near the user’s location or zip code. | 11.1) users can view what kinds of shifts are available near them depending on their location |

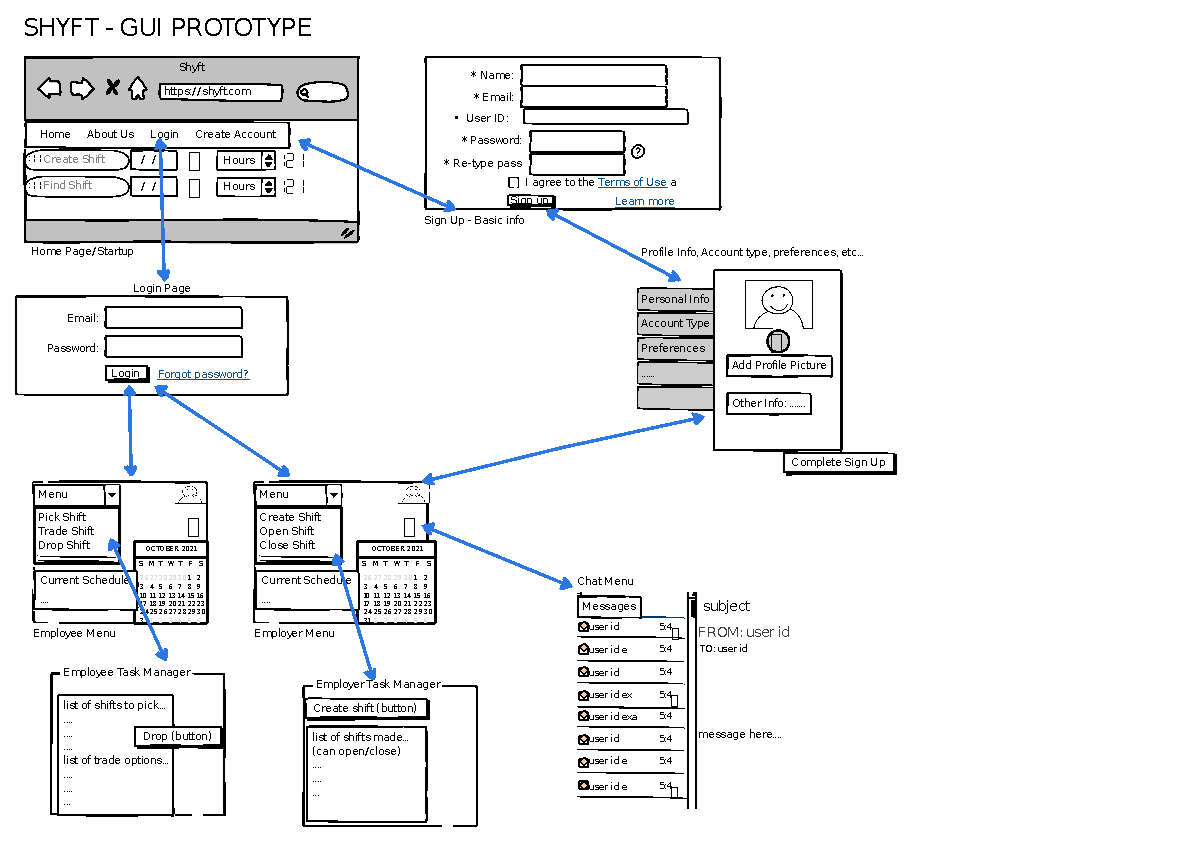
III. UI Mockups and Storyboards (high level only)

Picture 1: Gray and White Wire Diagram

James Giatpaiboon, Front-end Lead, Shyft

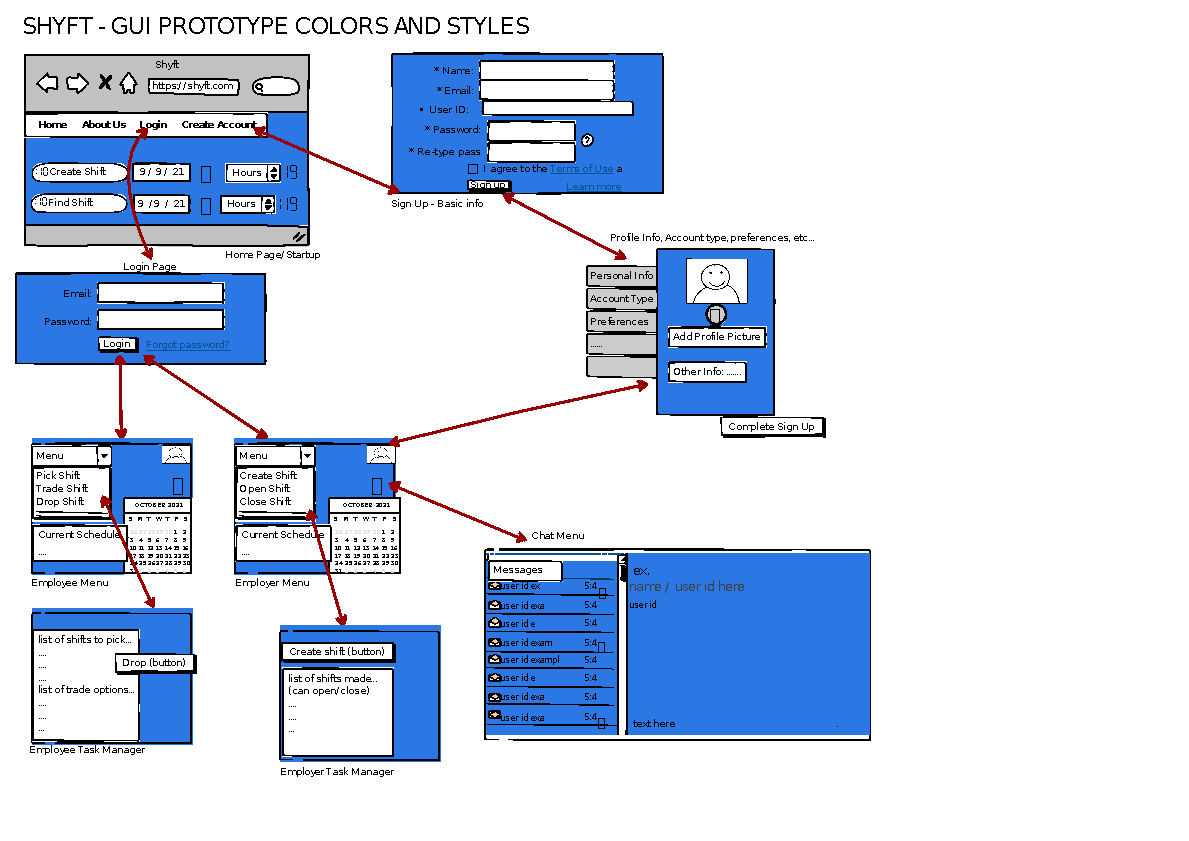
Picture 2: GUI Prototype for Shyft

(Sign-in/up, Create Shift, Pick Up Shift, Drop Shift)



James Giatpaiboon, Front-end Lead, Shyft

Picture 3: GUI Prototype for Shyft



James Giatpaiboon, Front-end Lead, Shyft

IV. High-level Architecture, Database Organization



User

| **Users** |
| --- |
| user\_id int PK |
| name varchar(255) |
| username varchar(255) |
| email varchar(255) |
| password varchar(255) |
| usertype int |

Resume

| **Resume** |
| --- |
| resume\_id int PK |
| user\_id FK -> users.user\_id |
| resume\_path varchar(4096) |

Job

| **Job** |
| --- |
| job\_id int PK |
| title varchar(255) |
| skills varchar(4096) |
| salary int |
| street\_address varchar(255) |
| city varchar(255) |

| **UserSkills** |
| --- |
| user\_skills PK |
| user\_id FK -> users.user\_id |
| skills varchar(4096) |

| **Shift** |
| --- |
| shift\_id int PK |
| job\_id int FK -> shift.shift\_id |
| user\_id FK -> users.user\_id |

| zipcode int |
| --- |
| type varchar(255) |
| user\_id FK-> users.user\_id |

*Add/Delete/Search architecture Functional Requirement*

**Add/Delete/Search for Users** When users register

**Add/Delete for UserSkills** When a user adds skills

**Add/Delete/Search for Resume** When a contractor uploads or deletes their resumes

**Add/Delete/Search for Job** When a employer adds or deletes a job

**Add/Delete/Search for JobApplications** When a user adds or deletes a job application

**Search/Display for JobApplications** When a contractor looks at the users that applied for  
 their job

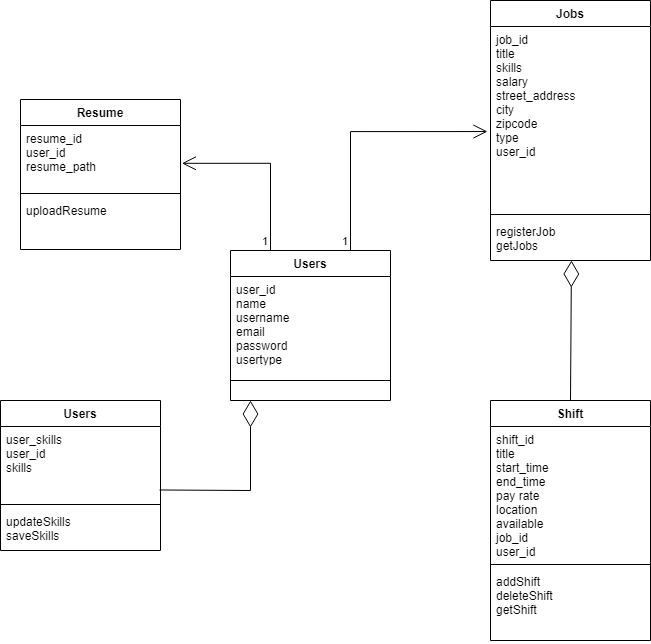
Technical Feasibility of DB operations

We considered the functional requirements for users when using database operations. An example would be when a userfilters out jobs based on skills. We would then look up the **Jobs** table, get any **Job\_ids’** that match the skills and display the jobs for the user to view and apply. When the user submits an application, a new row is added to the **JobApplications** table. Contractors can then view who applied to their job by getting the applications that match the **Job\_id** that they have selected.

*API’s*

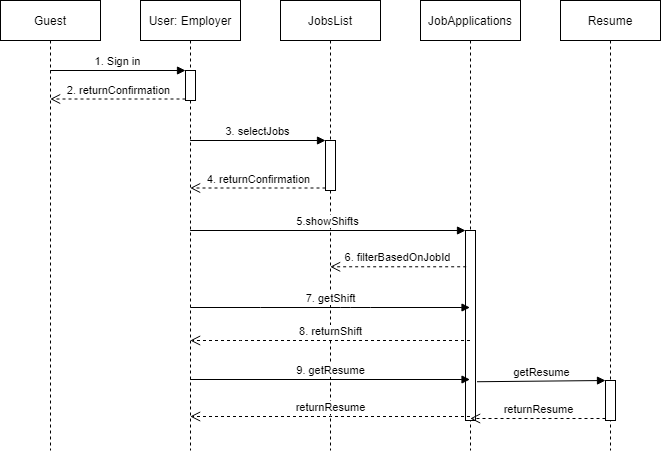
Our app will use an API created with express, allowing us to add, delete, and search through our database. We will also be using express on the backend and use axios to make calls to it.

V. High-level UML Diagrams



Andy Ouyang, Back-end Lead, Shyft

**High Level Sequence Diagram**



Andy Ouyang, Back-end Lead, Shyft

Resource: [Practical UML: A Hands-On Introduction for Developers (Embarcadero)](https://edn.embarcadero.com/article/31863)

VI. Identify actual key risks for your project at this time

* **Skills risks and mitigation plan**
  + Unfamiliarity with tech stack or APIs. This is a complex project with a lot of moving parts so we have to study/get familiar with our areas of responsibility or risk falling behind
    - **To resolve:**
      * Doing tutorial projects similar to the functionalities we are planning on implementing
      * Reaching out to mentors/classmates/professor/TA for suggestions/help
      * Striving to stick to with study schedules and sharing progress or resources together on Discord or during class meetings
  + Putting it all together. A lot of moving parts hard to get going together without major bugs/issues (front end + back end integration on AWS platform)
    - **To resolve:**
      * Start early and test rigorously to make sure main functionalities are implemented and working
      * Post updates on discord regularly in case someone else has to come in and help so they know what’s been tried/done
      * good GitHub comments/practice
  + GitHub issues working with 6 people
    - **To resolve:**
      * work on relevant branches (back end vs front end branches etc.)
      * ask early and when in doubt ask
      * do biweekly reviews/checkups on the repo
* **Schedule risks**
  + All of our teammates have heavy class schedules and have various obligations outside of school making it difficult to meet more than twice a week to coordinate collaboration sessions
    - **To resolve:**
      * engage more so on Discord and utilize Trello
      * Coordinate expectations/mini-deadlines for learning items
      * Create learning objectives relevant to each person’s sphere of responsibility
  + Procrastinating on deadlines
    - **To resolve:**
      * implement mini deadlines
      * more one on one meetings with team lead or relevant people for task
      * have some cushion between your work and the actual deadline time
* **Teamwork risks**
  + Not voicing out issues early, leaving problems to fester till the last minute
    - **To resolve:**
      * one on one check-ups with team lead/scrum master to ensure everyone is doing well and give an opportunity to voice concerns
      * Giving everyone space to work out their own spheres of responsibilities while encouraging asking for help if need be
  + Not communicating leading to repeat code and or broken functionalities
    - **To resolve:**
      * Having teammates report to respective leads for backend/frontend and have them delegate tasks
      * more one on one meetings with team lead or relevant people for task
      * Utilize Discord/Trello/GitHub repo logs to keep an eye on things and communicate with team

VII. Project management

We have divided the team into two smaller teams of three: front-end (FE) and back-end (BE). Our front-end team developed the client directory while the back-end team developed the server directory. Our BE lead was pivotal in implementing the database organization, server setup, and GET/POST functions.

We utilized Monday meetings to assign roles and tasks for the week. We held our weekly meetings outside of class on Fridays to provide end-of-week status updates for Milestone 2. We (TL and SM) also used this time to casually check in with each member and see how they are doing in general.

As a team lead, I found that starting early, planning ahead, and having contingency plans are critical in order to mitigate risks as we near a deadline. For future tasks, one front-end dev/ github manager suggested we create a visual workflow and manage the document in Google Docs. We will also continue using Zoom/Discord as the main medium of communication.