Software Requirements Specification

For Contractor Connect

PART A, B, C

Prepared by

Cibi Irin (301248509),

Liu Ziyan (301133339),

Mitsopoulos George (301181718),

Rana Smritika (301233977),

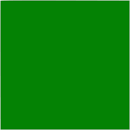
Wu Wilson (301243494),

Ramin Beiramzadegan(301254240),

Singh, Amritpal (3301236031)

COMP246 Group 1

2022-09-16

Highlights the edited part

Part A

Cibi Irin：1.1 & 6.1

Liu Ziyan ：2.1 & 3.4 & 4.1 & 4.2

Mitsopoulos George ：1.2 &1.3 & 5

Rana Smritika ：1.4 & 3.1 & 7.1

Wu Wilson : 3.2 & 6.2

Ramin Beiramzadegan: 8 & 9.1

Singh, Amritpal: 3.3 & 7.2

Part B

Cibi Irin：2.1 &2.2

Liu Ziyan ：3.1 & 3.2 & 3.3 & 4.1 & 9.1

Mitsopoulos George ：4.2.1

Rana Smritika ：4.3.1

Wu Wilson : 4.2.2

Ramin Beiramzadegan: 5

Singh, Amritpal: 4.3.2

Part C

Cibi Irin：1.1

Liu Ziyan ：1.2

Mitsopoulos George ：4.1

Rana Smritika ：1.3

Wu Wilson : 3.2

Ramin Beiramzadegan: 2.1

Singh, Amritpal: 3.1

Table of Contents

**Revision History**

**1.** **Problem statement............................................................................ 1**

1.1 a) Problem & Need 1

1.1 b) List of (i) capabilities and (ii) Benefits 1

1.2 Identify the stakeholders and their roles 1

1.3 Identify the sub-systems of your application ( What are its functional components) 1

1.4 Who are the intended users of the SRS documentation. 1

**2.** **A Context Flow – Structured Modeling..........................................................2**

2.1 Non-UML Context Flow Diagram – CFD 2

**3.** **Requirements – Functional -- UML Use Case Modeling...............................3**

3.1 Goal Use Cases 3

3.2 Use case Diagrams.......................................................................................................... 3

3.3 User Stories: 3

**4.** **UML Domain Class Diagram....................................................... ...................4**

4.1 List of the classes............................................................................................................ 4

4.2 Domain class diagram..................................................................................................... 4

**5. ERD.....................................................................................................................4**

5.1 ERD Diagram 4

**6.** **Two UML Systems Sequence diagram...........................................................4**

6.1 First Diagram 4

6.2 Second diagram...................................................................................................................... 4

**7. Two UML State Diagrams............................................................................** **4**

6.1 First Diagram 4

6.2 Second diagram....................................................................................................................... 4

**8.** **Techonologies** …................................................................................................................**4**

**9. Project Management..........................................................................................4**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| George | Sept\18\22 | Started work on 1.2 & 1.3 | 1.0 |
| Ziyan | Sept\19\22 | 2.1 | 1.0 |
| George | Sept\20\22 | Replaced temporary bullet points with a table, for 1.2 | 1.1 |
| Smritika Rana | Sept\20\22 | 1.4 | 1.0 |
| Irin Cibi | Sept\20\22 | 1.1 | 1.0 |
| Smritika Rana | Sept\22\22 | 3.1 | 1.1 |
| Ramin Beiramzadegan | sept\23\22 | 5,6 | 1.0 |
| Amritpal Singh | Sept\25\22 | Worked on 3.3 | 1.0 |
| George | Sept\26\22 | Added Payment services and government to the table for 1.2 | 1.2 |
| Ramin | Sept\27\22 | 4.1 | 1.0 |
| Wilson | Sept\27\22 | Added a use case diagram for 2 subsystems | 1.0 |
| Irin Cibi | Sept\28\22 | Made changes in 1.1) a | 1.0 |
| Wilson | Sept\28\22 | Merged goal use cases  Added 2 more use case diagrams | 1.1 |
| Ziyan | Sept\28\22 | Replaced 2.1 diagram | 1.1 |
| Irin Cibi | Sept \29\22 | Made changes in 1.1) b | 1.1 |
| Ziyan | Sept\29\22 | Updated 4.2 | 1.0 |
| Wilson | Sept\29\22 | Added more goal use cases on 3.1  Added UML System Sequence Diagrams | 1.2 |
| Irin Cibi | Oct \5\22 | Started on diagram 6.1 | 1.0 |
| Irin Cibi | Oct \12\22 | Updated diagram 6.1 | 1.0 |
| Wilson | Oct\15\22 | Updated goal use cases | 1.3 |
| Wilson | Oct\16\22 | Updated use case diagrams  Updated classes  Uploaded ERD | 1.1 |
| Ziyan | Nov\03\22 | Updated 1.2 & 1.3 & 2.1 & 4.2 & 7 | 1.3 |
| Part B |  |  |  |
| Irin Cibi | Nov\9\22 | Updated 2.1&2.2 | 1.0 |
| Ziyan | Nov\10\22 | Updated 3.1 & 3.2 & 3.3 & 4.1 | 1.0 |
| Ramin | Nov\11\22 | Updated 5.1 | 1.0 |
| George | Nov\18\22 | Started work on 4.2.1 Part B | 1.0 |
| Smritika Rana | Nov\19\22 | Started work on 4.3.2 | 1.0 |
| George | Nov\28\22 | Updated 4.2.1 Part B | 1.1 |
| Part C |  |  |  |
| Ziyan | Nov\29\22 | Updated 1.2 Part C | 1.0 |
| Ziyan | Dec\1\22 | Updated 3.2 &3.3 Part B | 1.1 |
| Smritika Rana | Dec\1\22 | Updated 1.3 | 1.0 |
| George M | Dec\6\22 | Started Part C 4.1 | 1.0 |

# Problem statement

## a) Problem & Need

According to the Pulse report, 4 out of every 5 Canadians work for more than 40 hours every week. This excludes the other chores and errands they must do in everyday life. People nowadays lead a very busy lifestyle, which makes it almost impossible for them to find the time for other chores. Everyone's in search of an easy way out that is also precise and efficient.

Moving can be very stressful as it is but with the additional work that needs to be done, like mounting a TV, hanging pictures on the wall, setting up the furniture, and so on, the workload builds. These tasks could be done by themselves, but the work won't be as perfect or precise as it is when done by a professional contractor. This is when Contractor Connect comes in handy. Contractor Connect is an online application that matches freelance labor with local demand, allowing customers to find help with everyday tasks such as handyman work, moves, and deliveries

Finding a good, trustworthy contractor to get the job done on time can be both hard and time-consuming. Contractor Connect lets you connect with over a thousand verified contractors that can get these chores done for you professionally. Nowadays everybody has a very busy lifestyle and finding the time to accommodate the contractor’s schedule can be very difficult. Therefore, our app lets customers align the contract work with their busy schedules by picking out a date and time of their convenience. Finding good, secure employment with benefits can be very difficult. Contractor Connect provides thousands of well-paid job opportunities to those looking for employment. After submitting your resume and registering on the app, each applicant will go through a verification process, after which if eligible they will be contacted through the app for several contract jobs at a time of your convenience. Most contractors prefer direct cash payments, which can be difficult since a vast majority of people don’t carry cash with them. Our app provides various payment methods that allow you to complete a transaction safely in a short span of time. Breach of data from web applications is a serious threat nowadays. Contractor Connect uses a firewall that protects it against malicious HTTP traffic. It places a filtration barrier between the targeted server and the attacker. This helps to protect all the information about our clients and contractors. Sometimes customers might not be aware of what type of contractor they might require. Therefore, customers can post pictures and a description of the problem on the app. This post will be viewed by our thousands of verified contractors who will reach out to you if they can resolve it.

* Can the actor take some other action at this point?
* Customers can find the contractor’s phone number through google and call them to make a reservation. Moreover, they could contact the contactor whom they are familiar with. If it is an easy task, customers could even fix it by themselves.
* Is it possible that the actor will encounter some error condition at this point?
* Customers could find an inappropriate or unprofessional contractor, more expensive services, unclear reservation time, unguaranteed working period, and unsatiated results.
* Is it possible that the actor will encounter some other behavior at this point (for example, behavior that is invoked by some event outside the actor’s control)?
* During the repair, if bad weather or other such conditions occur, the customer will pay extra money for the delayed work length.
* Can poor system performance result in unexpected or improper user actions?
* Customers would be impatient to deal with the system when they face an emergency. Furthermore, poor system design would fail in customers’ expectations, especially when they are looking for a professional contractor.
* What information does the actor desire from the system?
* The actors will be looking for the desired contractors and make a reservation at their preferred time.
* Does the actor wish to be informed about unexpected changes?
* Yes, if the contractor has a time conflict, the actor wishes to be informed. Moreover, if the working period takes a long time, and the complexity of the maintenance process is beyond the contractor’s expectations, customers wish to know and are offered better advice.
* What are the preconditions, triggers, exceptions, and open issues?
* The triggers will be broken stuff in the customer’s house, it could be walls, pipes, electricity, etc. Exceptions could be small damage which could be easily fixed or cheap materials which could be easily replaced.

## b) List of Capabilities and Benefits

1. **Capabilities**

* Select the required service
* Connect with contractors
* Schedule the work at a date and time convenient to the customer.
* Register as a contractor
* Payment systems such as debit cards, credit cards, and apple pay.
* Register as a customer to save preferences or requirements.
* Log into the app using existing Google or Facebook accounts.

1. **Benefits**

* The app has over a thousand contractors that have been individually verified by our team, so they are efficient and trustworthy.
* Customers can schedule the work at their convenience that aligns with their busy schedules.
* If customers are unaware of the reason behind a problem or the type of contractor required, they can post pictures and provide a description of it on the app, which will then be viewed by our contractors who will reach out to you.
* Contactor Connect uses a firewall to protect the client and contractor information.
* The app provides various payment methods making it easier for customers to make secure payments smoothly.
* The app provides secure and well-paid employment to thousands of contractors with benefits.

## Identify the stakeholders and their roles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stakeholder Position** | **Role** | **Internal or External** | **Interest Level** | **Operational or Executive** |
| Local & Nationwide Contractors | Made up of (Electricians, plumbers, & architects) waiting to be hired by a customer | External | High | Operational |
| Job Seekers | Electricians, plumbers, & architects looking for contractor companies to work at. | External | Medium | Operational |
| Customers | Looking to hire contractor(s) & leave a review | External | High | Operational |
| Web Developer | Add new features and keep the website up to date | Internal | High | Operational |
| Sysadmin | Responsible for handling the website’s servers | Internal | Medium | Operational |
| Marketing Team | Responsible for advertising the website | Internal | Medium | Executive |
| 3rd Party Payment Services | Responsible for transactions made on the website | External | High | Executive |
| Government | Responsible for ensuring the contractors are certified | External | High | Executive |
| Database Administrator | Responsible for manipulating data from customers and contractors | Internal | Medium | Operational |
| Manager | Responsible for distributing work for contractors | Internal | High | Executive |
| Insurance Company | Responsible for customers’ repair fees | External | Low | Executive |

|  |
| --- |
|  |

## Identify the sub-systems of your application (What are its functional components)

- Login & Signup page (Signup process requires a Username, Password, Email, & Password)

- Customizable Business and Customer Profile Page

- Direct Message (DM) Other Users

- Website Security

- Payment subsystem

|  |
| --- |
| * Contractor searching subsystem (Customer) * Job Posting subsystem (Freelancers) * Hiring subsystem (Contractors) * Distributing freelancers subsystems(manager) |

## Who are the intended users of the SRS documentation.

Internal users:

* Web developer
* Database Administrator
* Programmer
* IT security
* Marketing Team
* Sysadmin

External users:

* Customer
* Job seeker
* Contractor

# A Context Flow – Structured Modeling

## A Context Flow – Structured Modelling

Diagram

Description automatically generated

|  |
| --- |
|  |

# Requirements – Functional -- UML Use Case Modeling

## Goal Use Cases

* **Identification Subsystem**

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Goal use case | Role player | Description |
| FR01 | Create an account | General web user | Web users must create an account by signing up that requires username, password and Email. |
| FR02 | Log-in to the account | General web user | Web users must log-in to the account using username and password. |
| FR03 | Access using Google and Facebook accounts. | General web user | Web users can log-in to the account using existing Google and Facebook accounts. |
| FR04 | Reset password | General web user | The system must allow web users to create new passwords if the entered password is invalid. |
| FR5 | Delete account | General web user | Web users can close their account and delete it from the database if they choose |

* **Profile Subsystem**

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Goal use case | Role player | Description |
| FR06 | Customize profile | Contractors/Customers | The system must allow users to customize the profile as per requirements like nearby location, type of services provided/needed, and availability |
| FR07 | Enter personal details | Customers | The system must allow customers to create and edit the profile like address, payment method, and others. |
| FR08 | Display posts | Customers/Contractors | Users can choose which posts to display at the top of their profiles. |
| FR09 | Add finished projects | Contractors | Contractors will be able to add pictures and descriptions of past jobs to prove qualifications |

# Direct message subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Goal use case | Role player | Description |
| FR10 | DM other users | General web user | The system must allow users to chat or direct message other users like contractors, customers for any queries, private appointment setting and its detail. |
| FR11 | View messages from others | Customer. /Contractor | The system will have a separate area to view all messages received/sent in one place |
| FR12 | Save dates to calendar | Customer  /Contractor | The system will allow users to save an entry in their calendar for specific dates |

* **Contractor matching subsystem**

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Goal use case | Role player | Description |
| FR13 | Create matches between customers and contractors | Customer/Contractor | The customer's job request will go through the database and find appropriate contractors for the job |
| FR14 | Create a link between the customer and contractor so that discussions can be had | Customer/Contractor | The customer and contractor will be connected through the direct messaging system |
| FR15 | Create a contract | Customer/Contractor | Contractors will be able to fill out PDFs for contracts which will be sent to customers to sign through the app |
| FR16 | Create posts and requests | Customer | Customers will be able to post pictures and descriptions of jobs which need to be done |

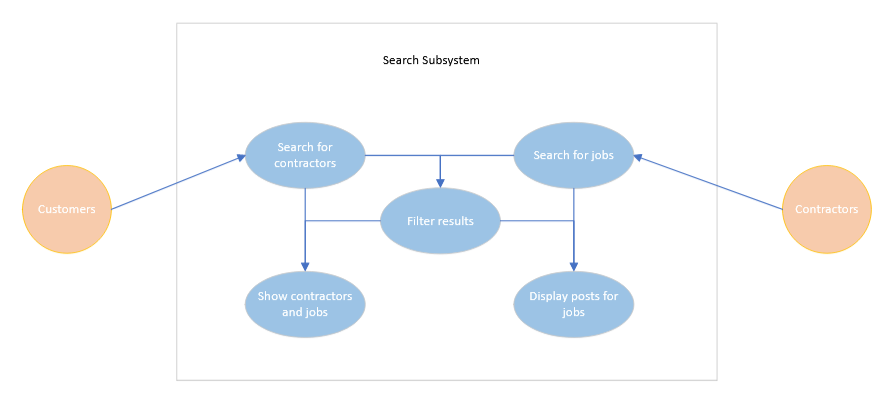
* **Search/filter subsystem**

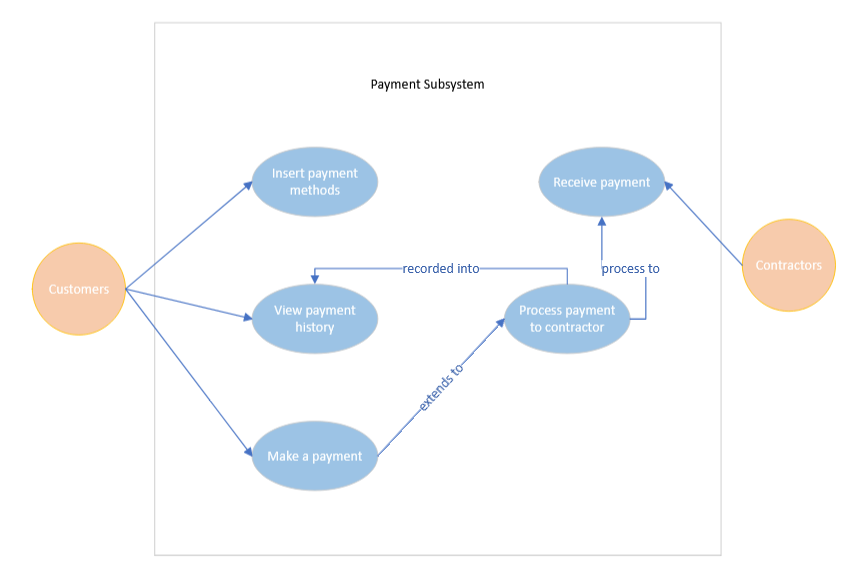
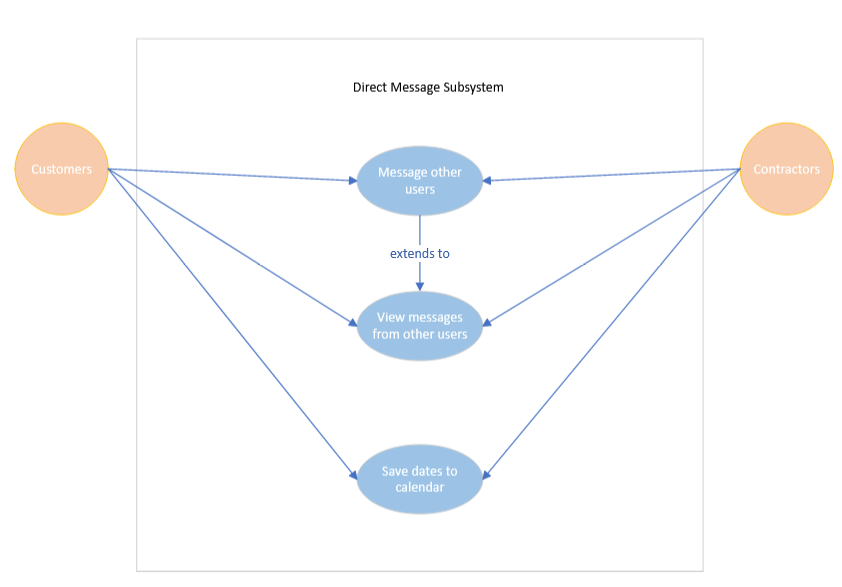
|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Goal use case | Role player | Description |
| FR17 | Search for contractors/ jobs that contractors do | Customers | The customer will be able to search for specific trades and view available contractors for the jobs |
| FR18 | Search for jobs that customers posted | Contractors | Contractors can search for customers looking for work to be done |
| FR19 | Different search system depending on type of account | Contractor/Customer | The type of account (Customer/Contractor) will change the search conditions so customers can only find contractors and vice versa. |
| FR20 | Filter search by different parameters | Customer/Contractor | The user can use a filter bar to search for most relevant, by age, or jobs by price/pay |

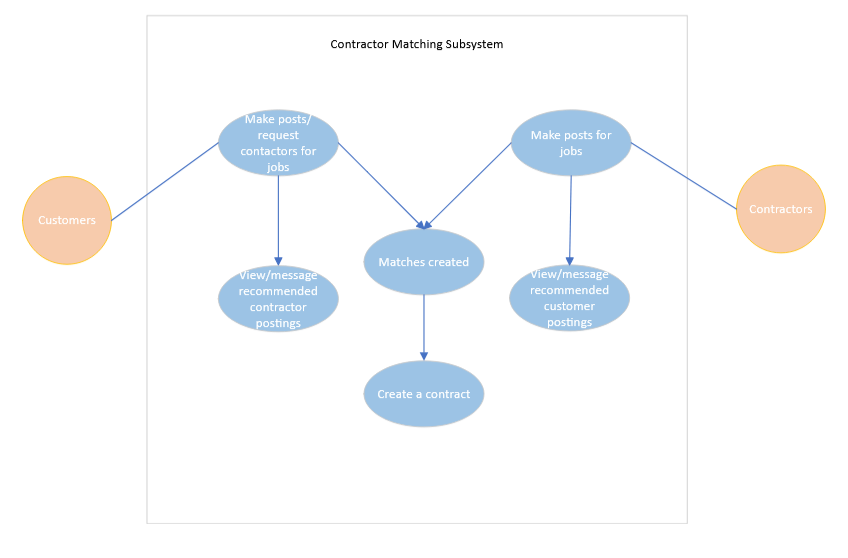
* Payment Subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Goal use case | Role player | Description |
| FR21 | Update the payment method | General web user | The system must allow the users to add payment methods with additional options like a third-party portal. |
| FR22 | View payment history | General web user | The system must allow the users to see the payment history and billing details. |
| FR23 | Receive payment | Contractor | The system will allow a payment to go through the app and be deposited directly into the contractor's bank account after fees |
| FR24 | Make a payment | Customer | The system will have a payment option which will allow users to book a contractor's services directly through the website |

## Use case diagrams







## User Stories

**3.3.1 Access Management Subsystem**

As a user, I want to create an account so that I can connect to the Contractor Connect app.

Acceptance Criteria:

* Should be able to select account type (e.g., Contractor, or Customer).
* Should be able to create a valid username and password.
* Must enter a valid email which is not registered in the past.
* Users will receive a registration confirmation mail at the end of the registration process.
* Form can only be submitted if all the required fields are filled.

As a user, I want to log in to the app so that I can access the Contractor Connect services.

Acceptance Criteria:

* Enter a Valid username and password to log in into a customer or contractor account.
* Access granted to app services after entering valid login details.
* User taken to login page upon opening app.
* User taken to home page after entering valid login details.

As a user, I want to update my account so that I can change any information about requirements to reflect my current details.

Acceptance Criteria:

* Should allow the user to change information in their account (e.g., address, phone number, contract type etc.)
* Should not allow changing particular information such as username without validating user.
* Changing security-related details (e.g., email or password) requires inputting current password.
* Any new login and changes will be notified by email.
* Clicking the submit button will make the changes permanent until further changes.

**3.3.2 Appointment booking Subsystem**

As a House owner, I want to book an appointment so that I can access a contractor for handy work.

Acceptance Criteria:

* Past dates or hours cannot be chosen.
* From the list of possibilities, the user can select a date and time.
* A "Select Contractor" button that displays the "Search for a Contractor" use case allows the user to select a contractor.
* After booking is complete, user will get an email with the appointment information.
* A button labelled "Confirm Contractor Booking" completes the booking.

As a user, I want to be able to reschedule/cancel an appointment so that I can adapt to the changes in my schedule.

Acceptance Criteria:

* Cannot select dates or times in the past
* Changes to an appointment notifies all parties involved (email, app notification, etc.)
* Canceling frees up the time slot for other users to select
* Changes are finalized through a submit button

**3.3.3 Payments Subsystem**

As a contractor, I want to create an invoice so that I can bill clients.

Acceptance Criteria:

* Must be able to fill out invoice information.
* Should be in a position to confirm the total amount due.
* Should be able to use the app to submit the invoice to the clients.
* Clients should be able to access it using their payment details.

As a Customer, I want to store payment data so that I can easily make new payments.

Acceptance Criteria:

* Should be able to confirm if the app will keep payment information.
* Should be possible to see the sums of prior payments.
* Must be able to see invoice information.

As a contractor or customer, I want to view payment history so that I can keep track of previous payments.

Acceptance Criteria:

* Should be able to use the app to access the payment history.
* By default, the payment history page displays the current year.
* Must have access to the complete payment history.
* Filtering the results based on different criteria needs to be possible.

As a customer, I want to access the payment portal so that I can make a payment.

Acceptance Criteria:

* Should be able to use a payment button or link to reach the payment portal.
* A secure payment system should allow for the entry of card details.
* It should be possible to save credit card data for the next payments.

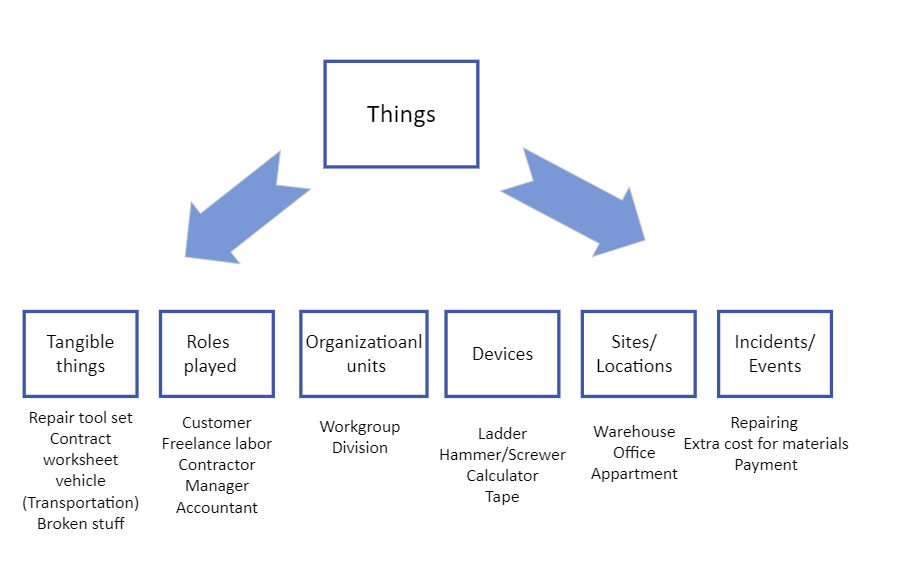
# System Features

**4.1 UML Domain Class Diagram**

4.1 list of classes

* Freelancers
* Services list
* Customers
* Pricing
* Invoice
* Rating
* Notification
* Message
* Order
* Appointment
* Account

**4.2.1BrainStorming**



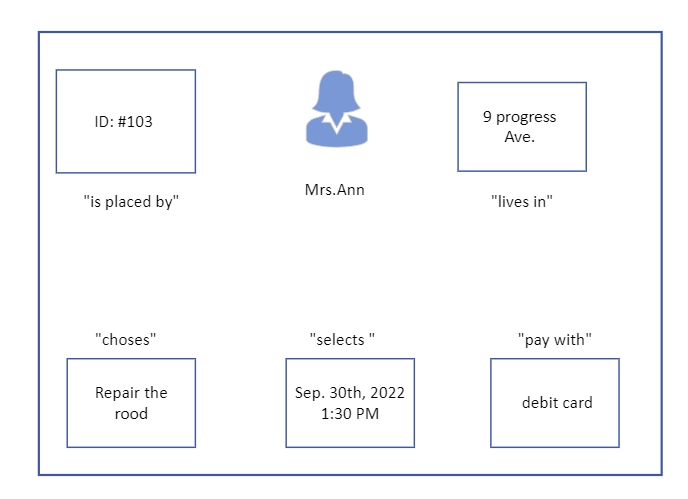
**4.2.2 Partial List of Nouns**

|  |  |  |
| --- | --- | --- |
| **Identified Noun** | **Notes on including noun as a thing to store** | **Store(Y/N)** |
| Customer | Does not matter with who are we working with | No |
| Confirmation | Checked or not | Yes |
| Transaction | Synonym as payment method | No |
| Change Request | Decline order by customer or contractor | Yes |
| Summary Report | Automatically summarized by the workers at the end of work | No |
| Worker | We know who they are | Yes |
| Backup Plan | No Backup Plan for cancellation of the order | No |
| Price | Depends on what type of work | Yes |
| Style | (One person or a group of people)  Decision-making | Yes |
| Payment method | Either pay in-person or online | Yes |
| Address for repairing | Distance is important | Yes |

4.2.3

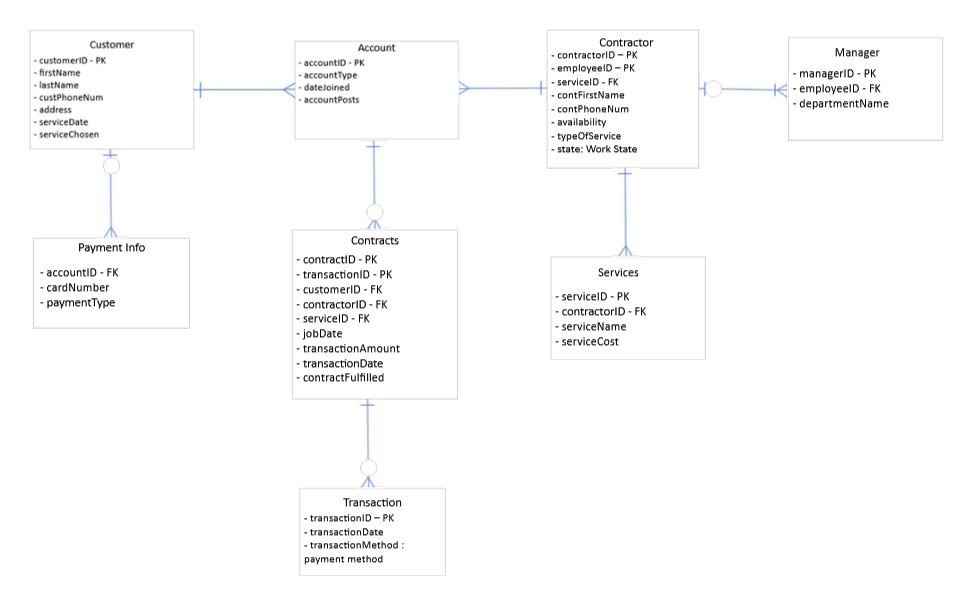
|  |  |
| --- | --- |
| **Class** | **Attribute** |
| Customer | - customerID - PK  - firstName  - lastName  - custPhoneNum  - address  - serviceDate  - serviceChosen |
| Contractor | - contractorID – PK  - employeeID – PK  - serviceID - FK  - contFirstName  - contPhoneNum  - availability  - typeOfService  - state: Work State |
| Account | - accountID - PK  - accountType  - dateJoined  - accountPosts |
| Transaction | transactionID – PK  transactionDate  transactionMethod : payment method |
| Manager | - workerID  - employeeID - FK  - departmentName  - managerID - PK |
| Work State | - work in process  - not working  - rest |
| Payment Method | - online  - in-person |
| Services | serviceName  serviceID - PK  serviceCost  ContractorID - FK |
| Payment information | accountID – FK  cardNumber  paymentType |
| Contracts | contractID - PK  customerID – FK  contractorID – FK  serviceID - FK  jobDate  transactionAmount  transactionDate  contractFulfilled |

**4.2.4 Associations among things**



|  |
| --- |
| Contactor Searching subsystem Domain Class Diagram |
| Job Posting & Hiring subsystem Domain Class Diagram |
| Distributing workers subsystem Domain Class Diagram |
| Full Domain Class Diagram |

**5. ERD Diagram**

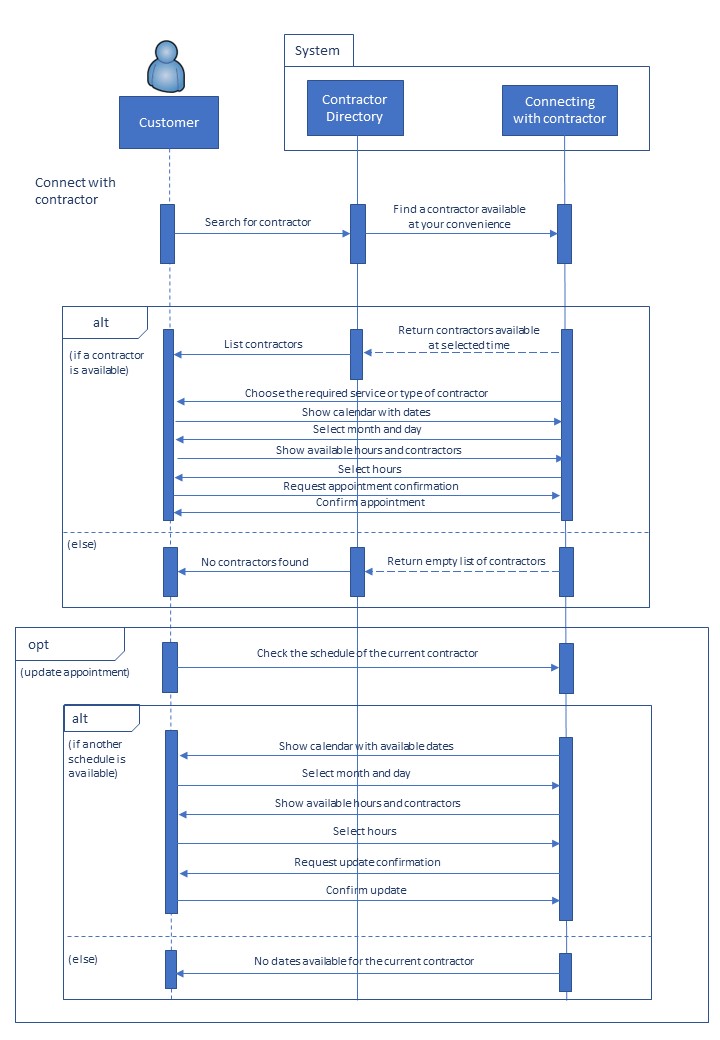
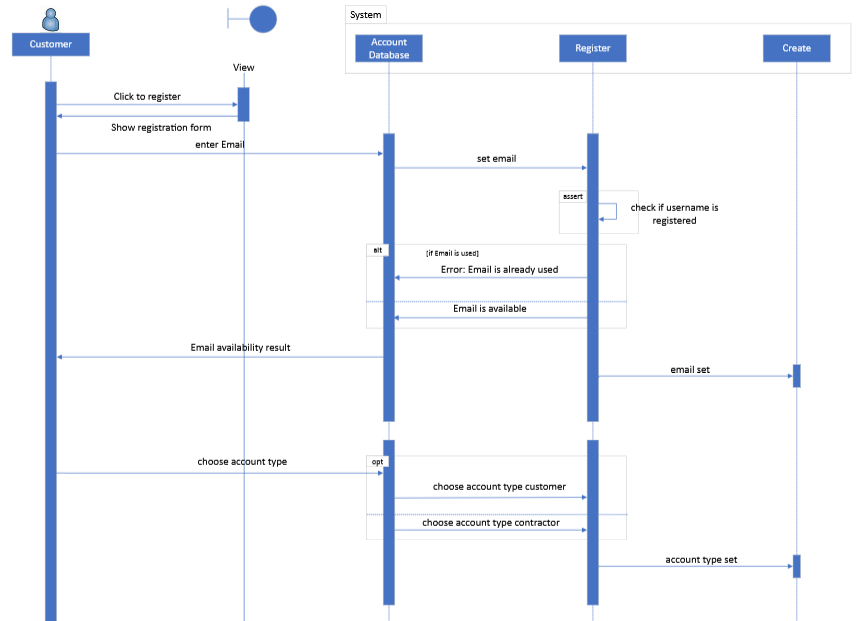


**6. UML System Sequence diagrams**

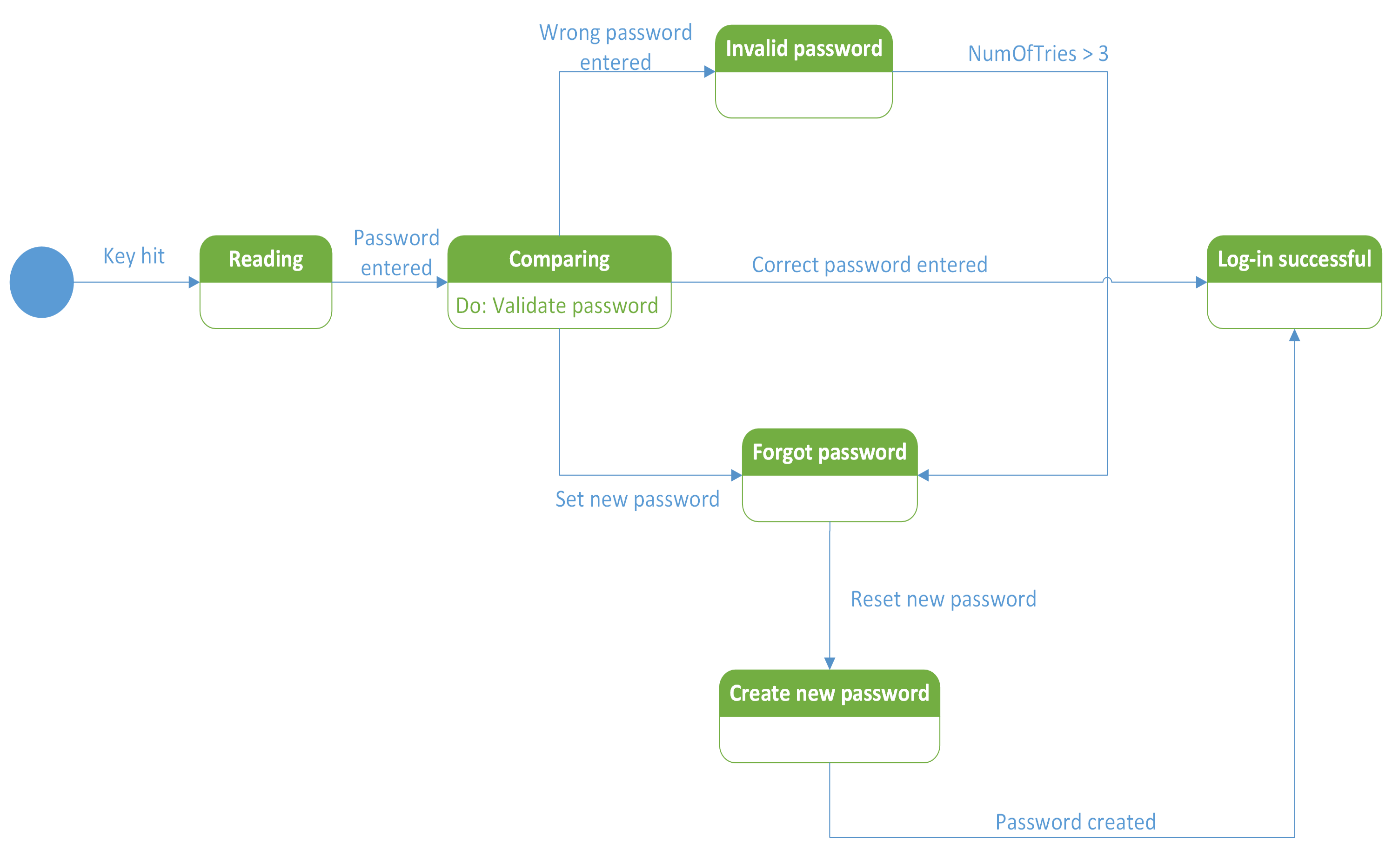
**6.1 System Sequence Diagram for Connecting Customer and Contractor Use Case**

**A picture containing timeline

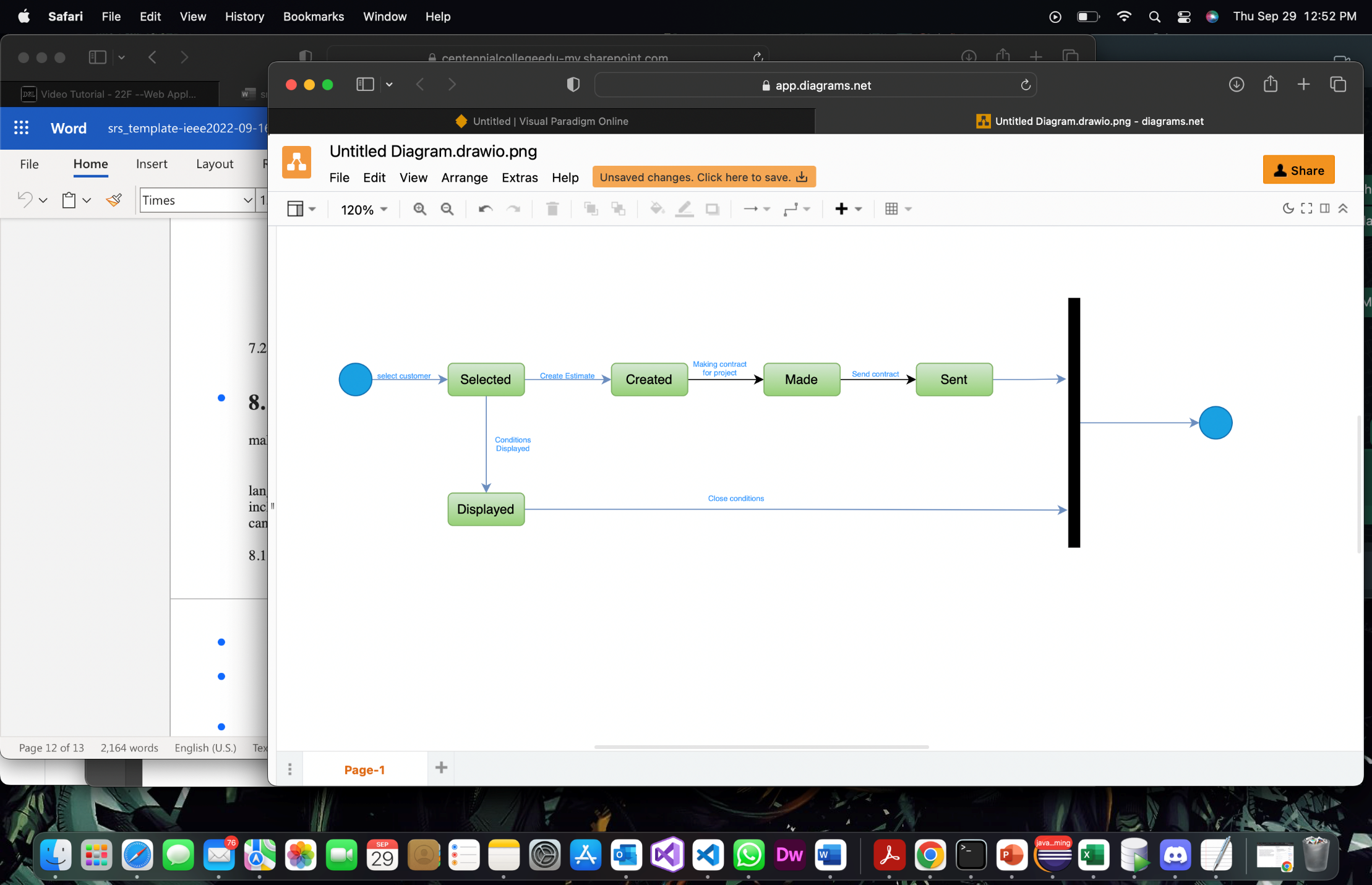
Description automatically generated**



**7. State Diagrams**

**7.1 State diagram for Account class**

**7.2 State Diagram For Finding Contract**



State Diagram for completing an order

**8. Technologies**

Contractor Connect is a website to link many freelancers to demanders or applicants and by this way makes it more accessible and more feasible.

In this website, we are going to use the Python as one of the most powerful programming languages and other web-service tools whether for client-side sections or server-side sections including JavaScript, HTML and CSS for front-end part and Django, MySQL for backend which can serve the users one the online platform.

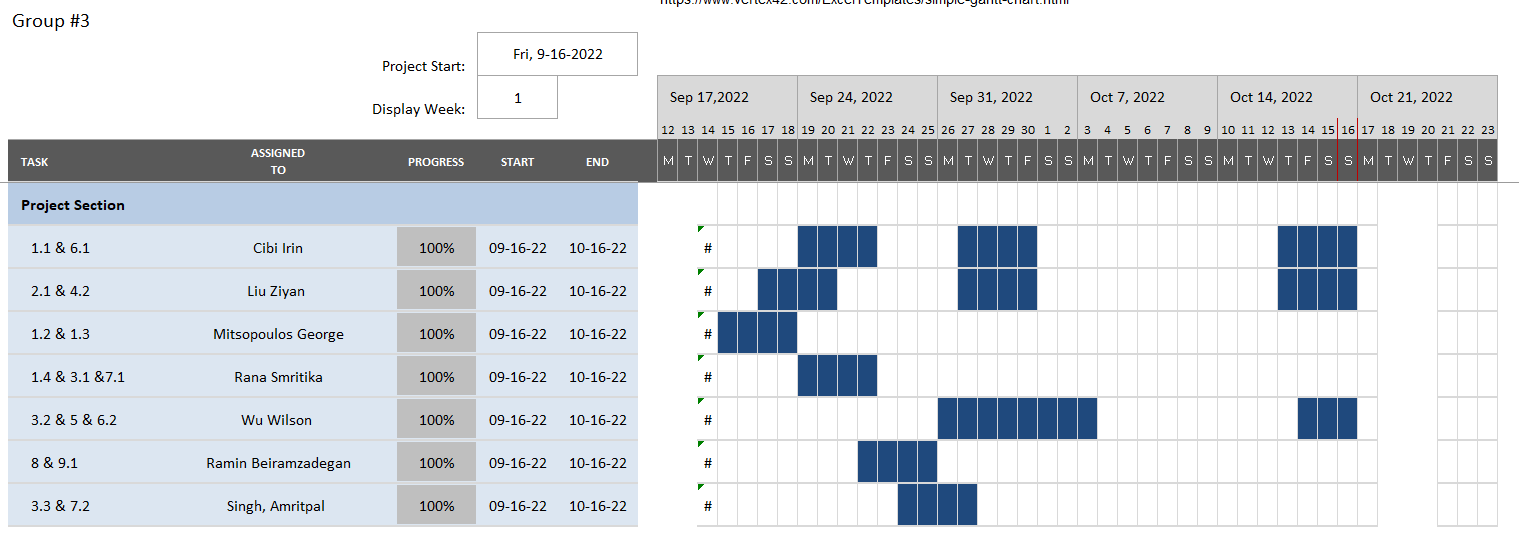
8.1 Client-side (Front-End, GUI)

* Bootstrap: the most popular framework of CSS to design our landing page or other internal web pages and decorating.
* https://getbootrap.com/
* HTML5: Hyper Text Mark Up language used for creating the elements of our website.
* Photoshop XD: is used for UI/UX design, our main template before going for coding by developers must be designed graphically in this part.
* https://www.adobe.com/ca/products/xd.html
* JavaScript: a scripting or programming language that allows you to implement complex features on web pages

8.2 Server-side (Back-end)

* Django: the most powerful framework of python programming language
* https://www.python.org/
* MySQL: is the most adaptable database environment to Django
* https://www.mysql.com/
* Redex pal: an online tool to create secure web addresses
* https://regexr.com/
* FileZilla: to upload your website files
* <https://filezilla-project.org/>
* Django REST framework: is a powerful and flexible toolkit for building Web APIs. To connect our website with external web services.
* https://www.django-rest-framework.org/

**9**



--------

**Part B**

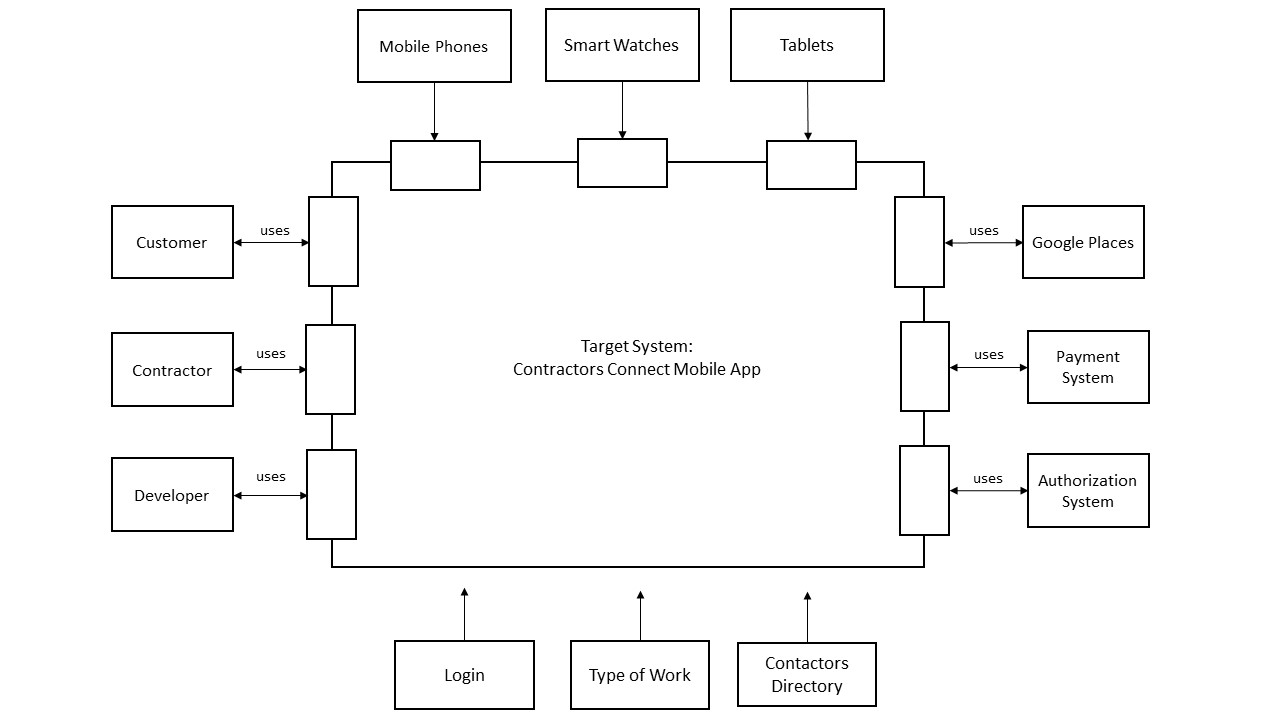
**2. Overview Model**

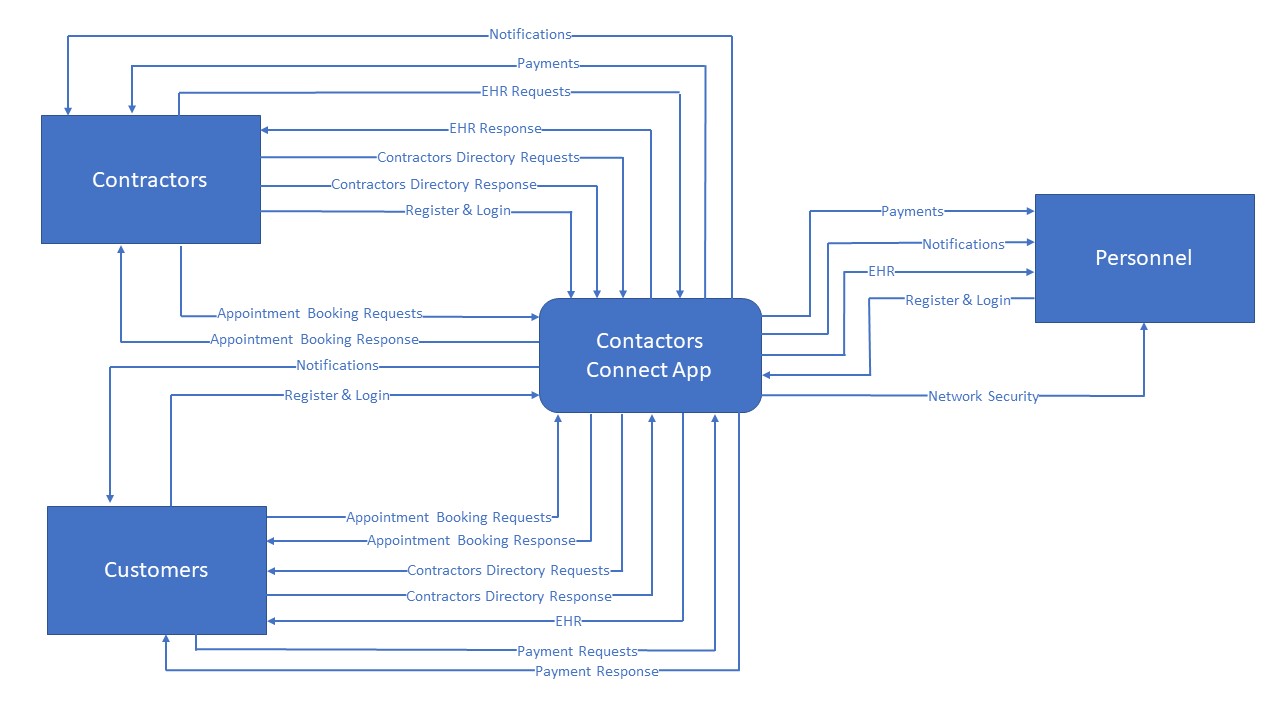
**2. Who are the intended users of the SDD document**

* Business Analysts
* Software Developers
* Quality Assurance Engineers
* Project Managers
* Stakeholders

The software design document should be a readable, understandable guide that describes the overall architecture of a software product. It contains information on a product’s functionality and the team’s goals. The SDD document is primarily used by the developers and stakeholders. In essence, software developers and QA engineers are the primary intended users of this document. To ensure the design meets all explicit and implicit requirements contained in the requirements model, business analysts should also consult this SDD.

**Part B**

**2.2 Architectural Context diagram**



**3. Modularization**

**3.1 Partition the analysis model into design components/subsystems.**

*Contactor Searching subsystem Domain Class*

**Freelance Registration** will be tasked with Sign-up and Sign-in

**Customer Registration** will be tasked with Sign-up and Sign-in

**Customer Account** will be tasked with account information

**Service cart** will be tasked with service information, contractor’s information, accepting and processing orders.

**Available Schedule** will be tasked with processing both contractor and customer’s schedule information and store schedule information.

**Payment Method** will be tasked with verifying payment from an external credit entity

*Job Posting & Hiring subsystem Domain Class*

**Hiring status** will be tasked with accessing contractor’s approvement and freelances’ status

**Contractor Account** will be tasked with account information and freelancer’s account information

**Job Posting** will be tasked with Application submissions and processing applications.

**Application** will be tasked with resume templates

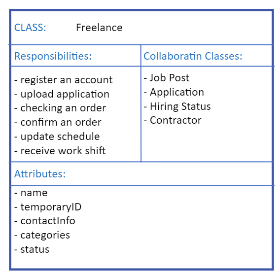
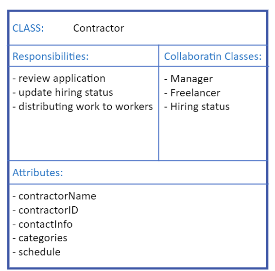
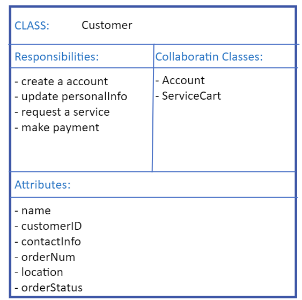
*Distributing workers subsystem Domain Class*

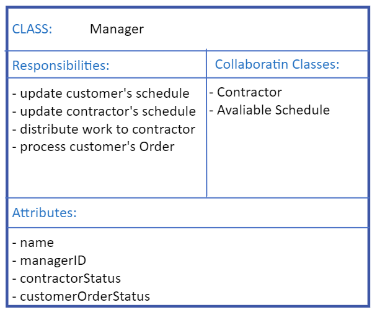
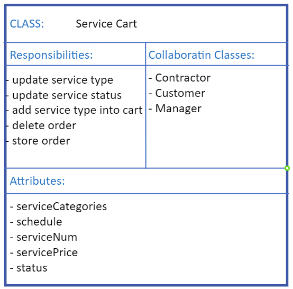
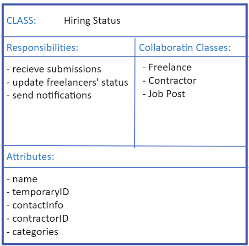
**Worker schedule** will be tasked with schedule assignments and accept/cancel button

**Contractor Account** will be tasked with accepting freelances’ resume and distribution of workers’ tasks.

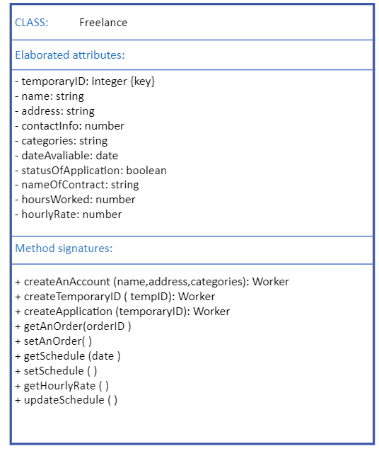
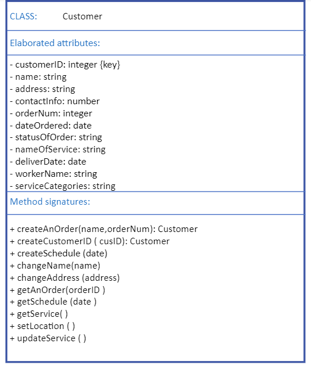
**Manager Account** will be tasked with accepting customers’ request and distribution of contractor’s work

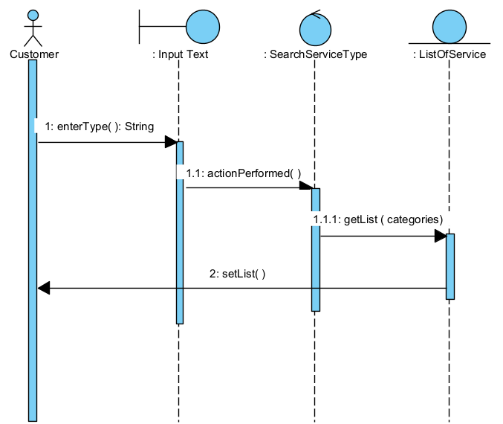
|  |
| --- |
| **3.2.1 First-Cut Design Class Diagram**   * #1 Use case: A customer selects a freelancer from service cart     **3.2.2 CRC**   * #1 Use case: A customer selects a freelancer from service cart     **3.2.3 FULL First-Cut Design Class Diagram**   * #1 Use case: A customer selects a freelancer from service cart     **4.1 Package MVC Diagram** |

**~~3.2.2 CRC card~~**

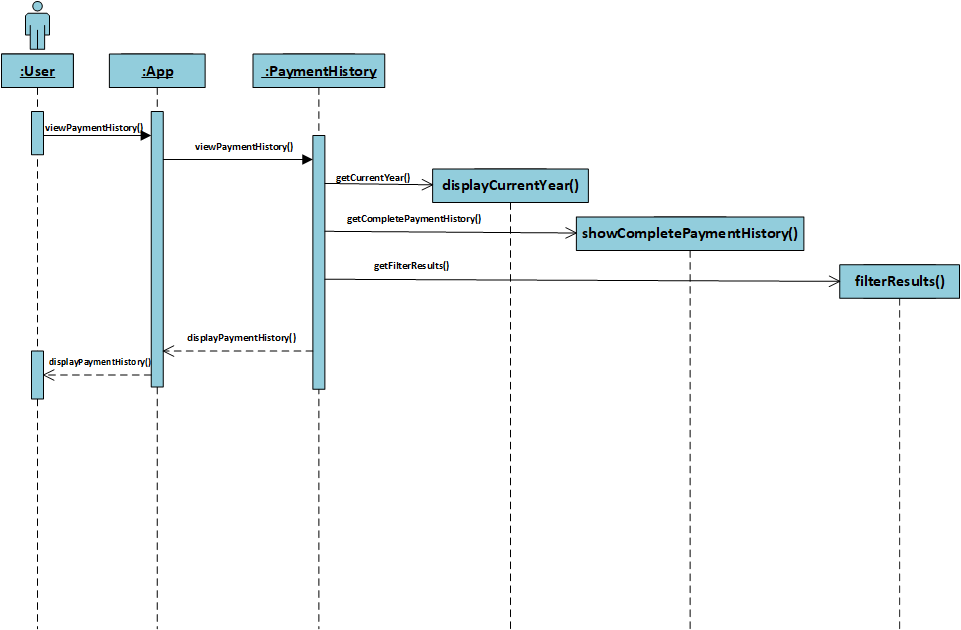


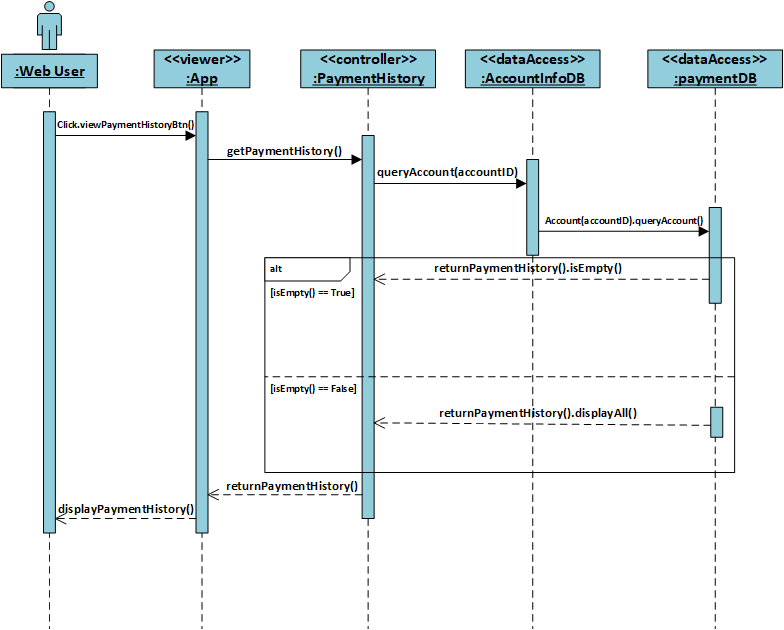
**~~3.3 Design class Diagram~~**



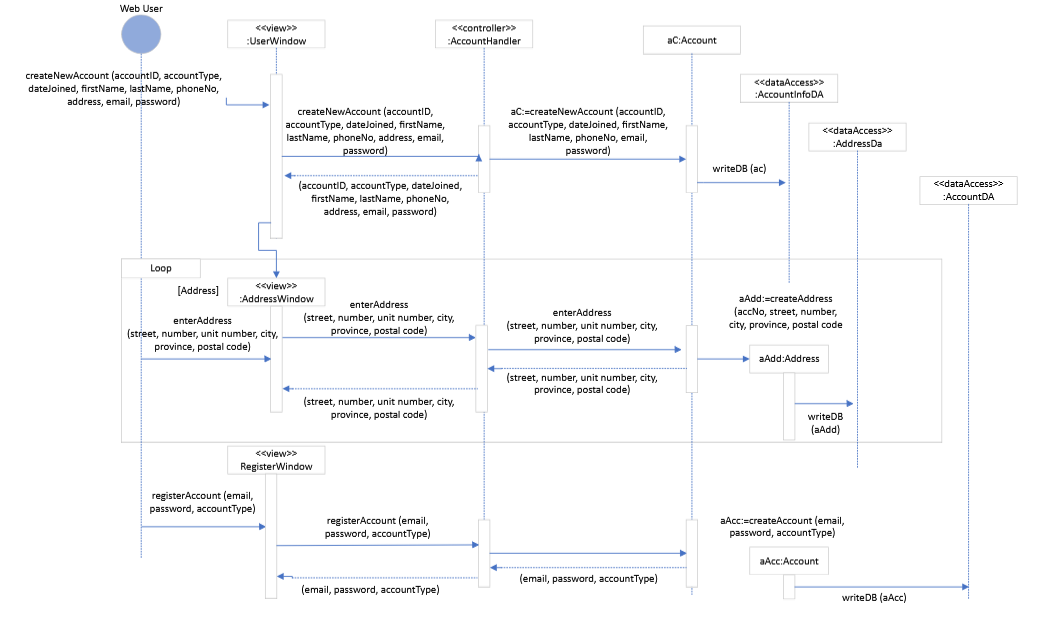
**~~4.1 MVC pattern diagram~~**

**4.2.1 One full sequence diagram of goal use case**

**Goal Use Case: View Payment History**

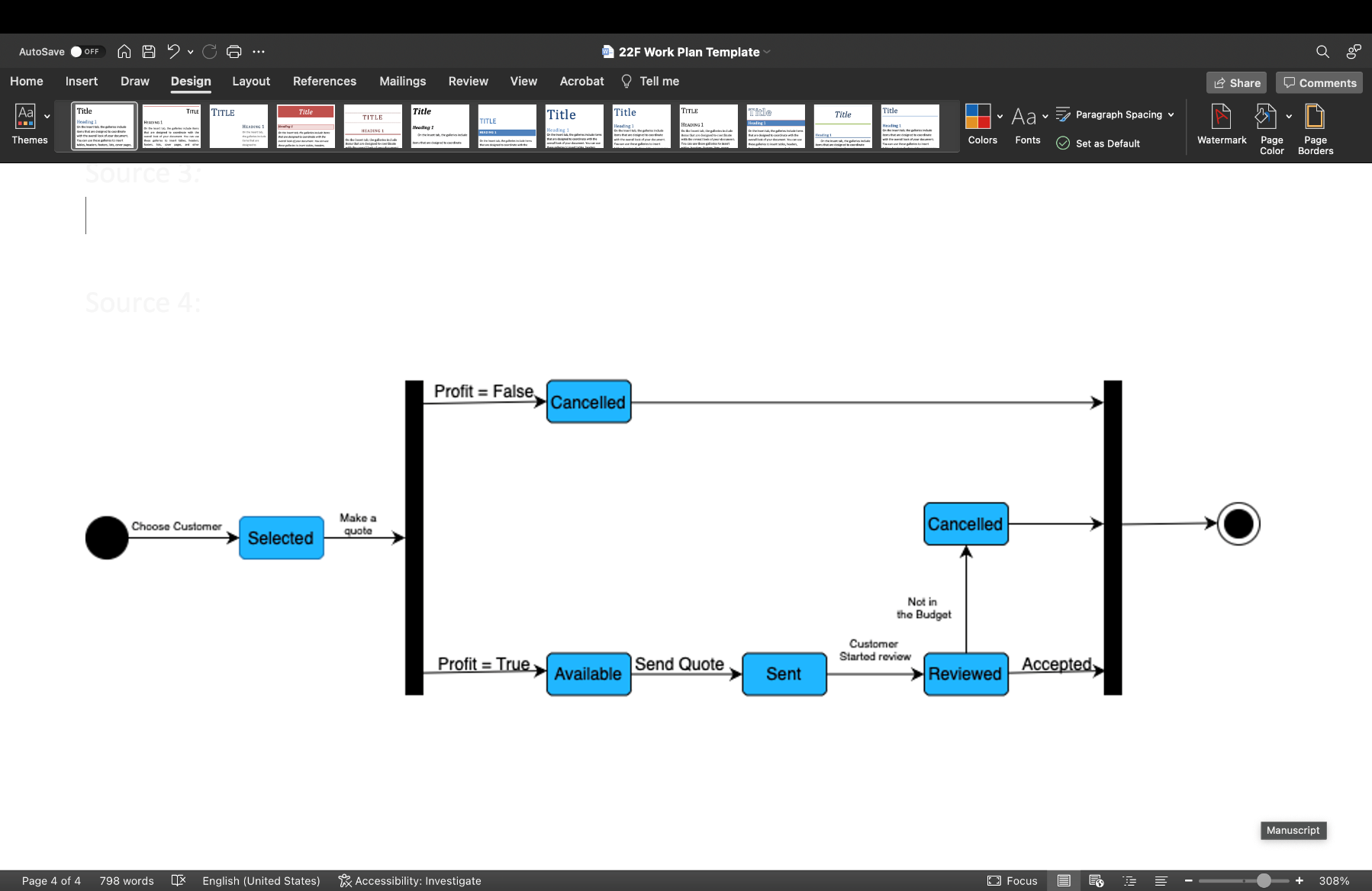


**4.2.2 One full sequence diagram of goal use case**

**Goal use case: Create an account**

**4.3.1 One state machine diagram of an object**

**State diagram for Account class**



**4.3.2 One state machine diagram of an object**

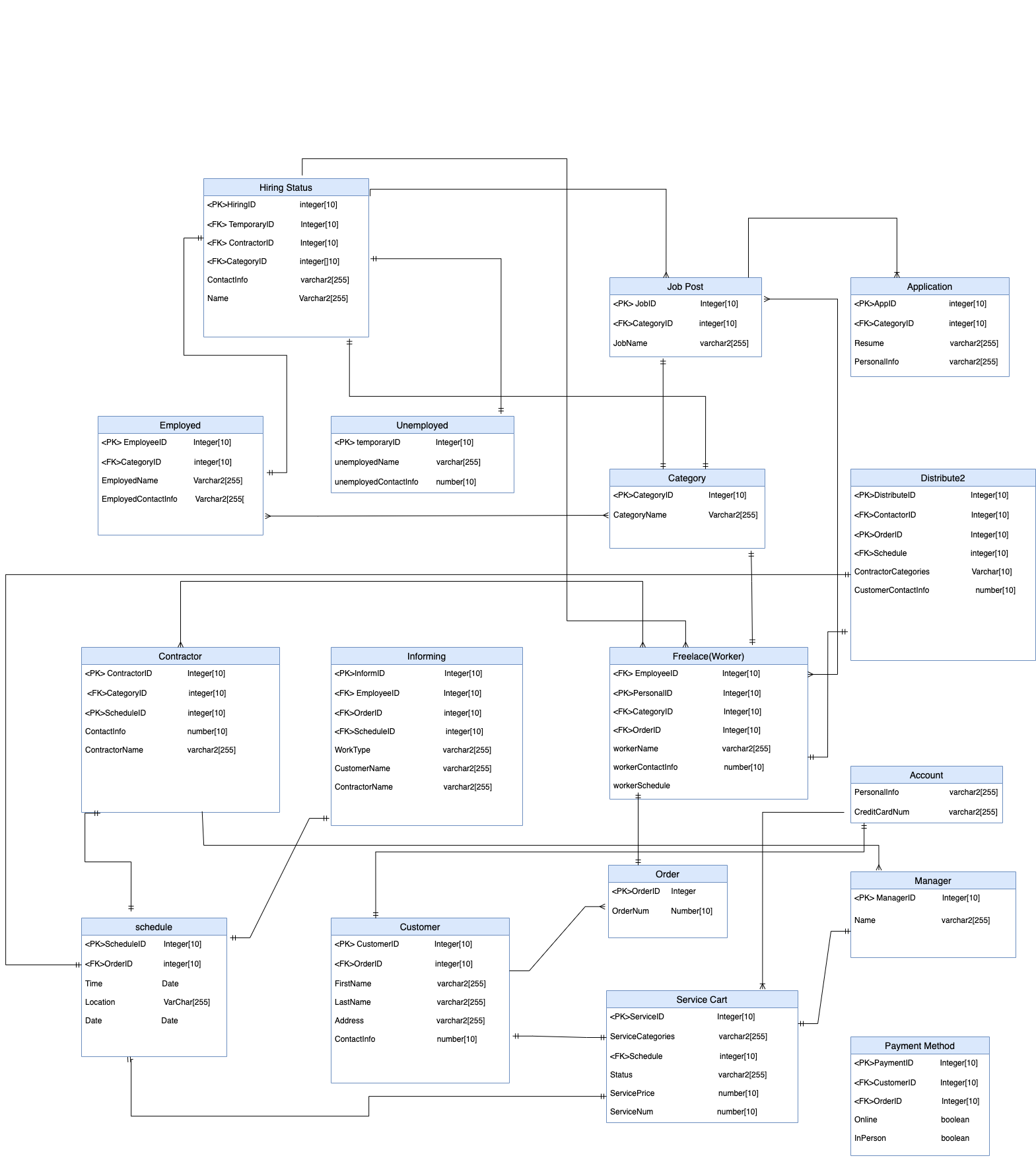
**State diagram for Appointment class**

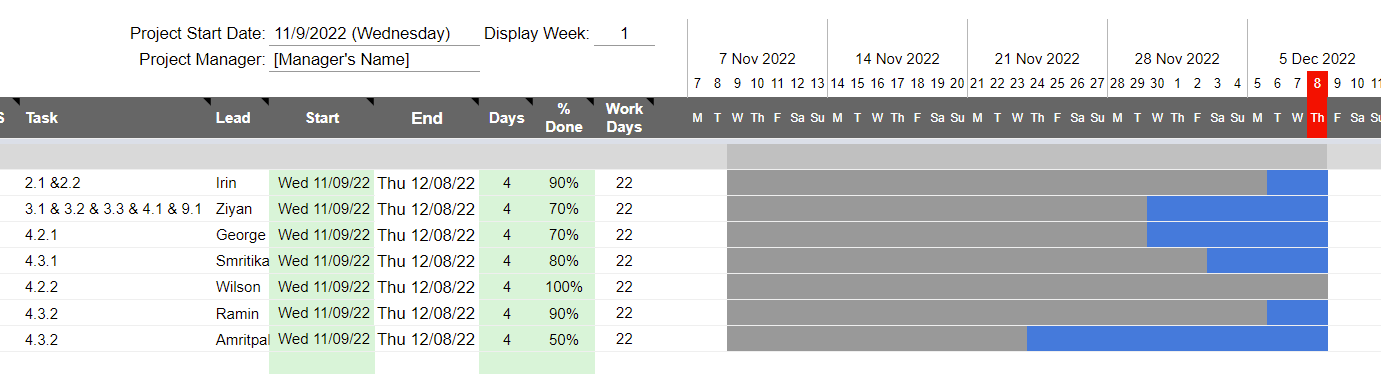
Diagram

Description automatically generated



**5.1 database schema**



Gantt Chart

**Part C**

**1.1 Monitor Design Patterns**

|  |  |
| --- | --- |
| Name: | Monitor |
| Problem: | Scheduled contract works might suffer some changes like the duration of the session, the scheduled time, or the assigned contractor. These changes need to be quickly informed to all users so that they can accommodate the changes or make a different appointment. |
| Solution: | ContractorConnect uses an observer pattern that alerts the user of any change in their appointments, with the use of an AppointmentBus that posts Appointments and are received by the users, therefore, keeping them informed of any important changes. |
| Example: |  |
| Benefits and Consequences: | With the monitor pattern, the users are now easily notified of the change in the schedule of their contract work, while also allowing them to send data to many other objects in a very efficient manner. |

**1.2 Singleton Design Patterns**

Service Cart

|  |  |
| --- | --- |
| Name: | Singleton |
| Problem: | Only one service cart can be created by a customer. The service cart class can be called when a customer selects a freelancer from the service list. Customer can delete or add an item in the service cart. This class should be encapsulated as “just-in-time" class |
| Solution: | Create a private service cart attribute, once the customer logs in, two scenarios will happen. Firstly, the customer has already picked the item and added it to the service cart. Second, if the customer has not added anything yet, the service cart will automatically appear and shows an empty cart. |
| Example: | *Class ServiceCart*  *{*  *Private static ServiceCart cart=null;*  *Public static getServiceCart()*  *{*  *If(cart == null ){*  *Cart = new ServiceCart();}*  *return cart;*  *}*  *}* |
| Benefits and Consequences: | The benefit of doing this singleton is to save more space for storage and create faster connections through encapsulation. The customer will receive a “just-in time” response. |

**1.3 Adapter Design Pattern**

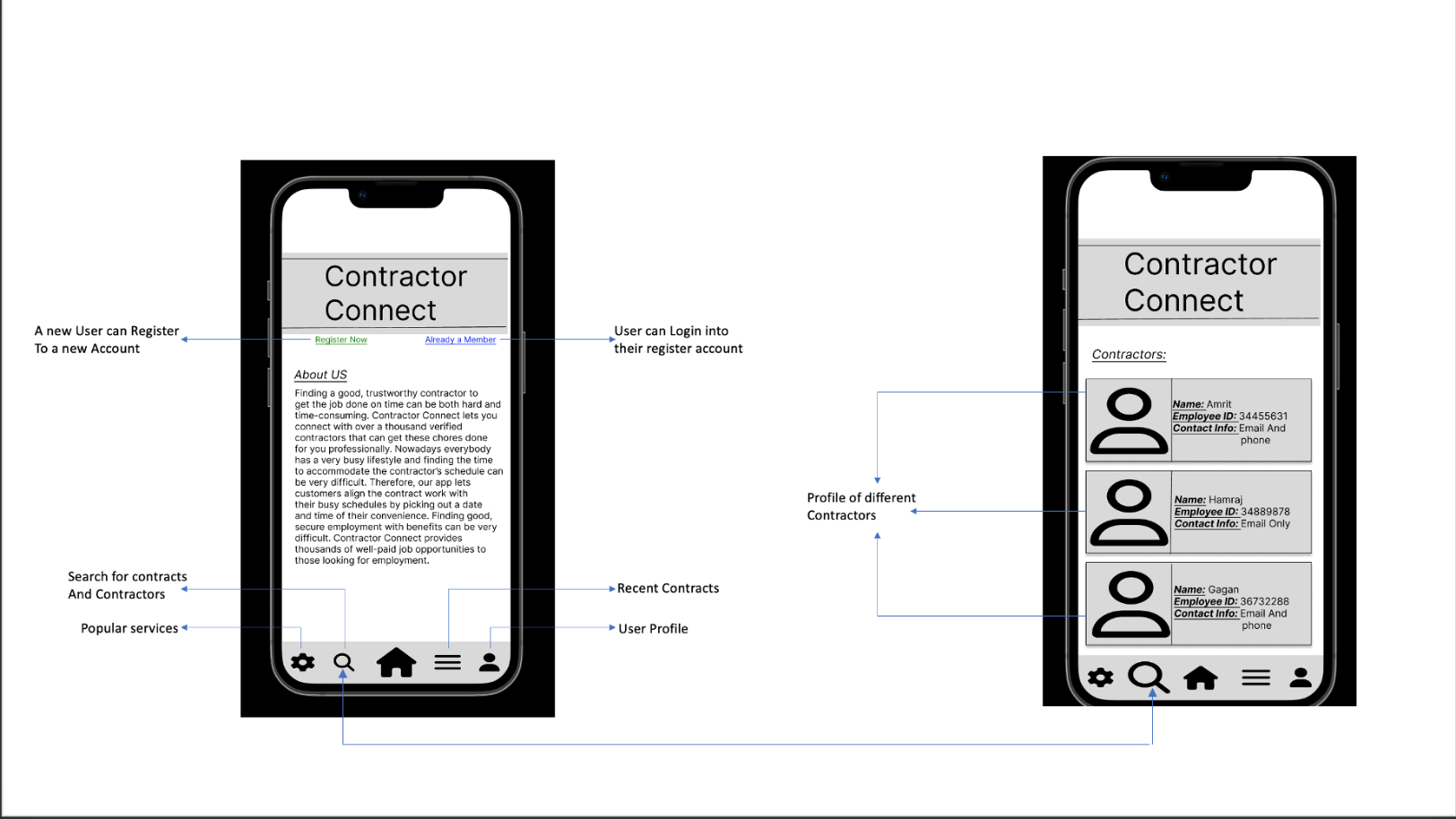
Service Cart

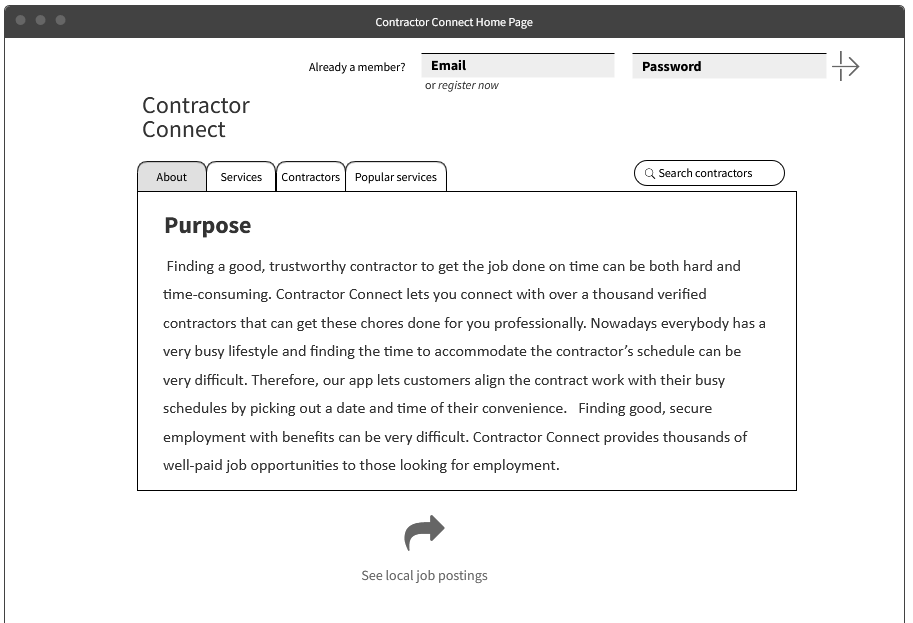
|  |  |
| --- | --- |
| Name: | Adapter |
| Problem: | The pattern that converts the interface of the class into another or joins functionalities of incompatible interfaces is known as adapter design pattern. Customer use service cart to request service. It needs another new class that can replace the original class with correct desired method signatures. |
| Solution: | Adapter class serves a link connection between the original class and class replaced. Here, the adapter class Services has the same method signature as the original class i.e., serviceCart. Customer can request the needed service through Services adapter as same as in Service cart. |
| Example: |  |
| Benefits and Consequences: | Customer can easily request the services through service cart or services i.e., adapter. It allows proper communication between the interfaces. |

2-1 Pattern Organization Table - Ramin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Design Patterns | | |
| creational | structural | behavioral |
| Subsystem  Components | Service Cart | singleton | adapter | State |
| informing | factory | proxy | Observer, state |
| Freelancers | Factory, singleton | adapter | observer |
| Scheduling | singleton | proxy | observer |
| User Interface | Singleton | Facade | Observer, State |

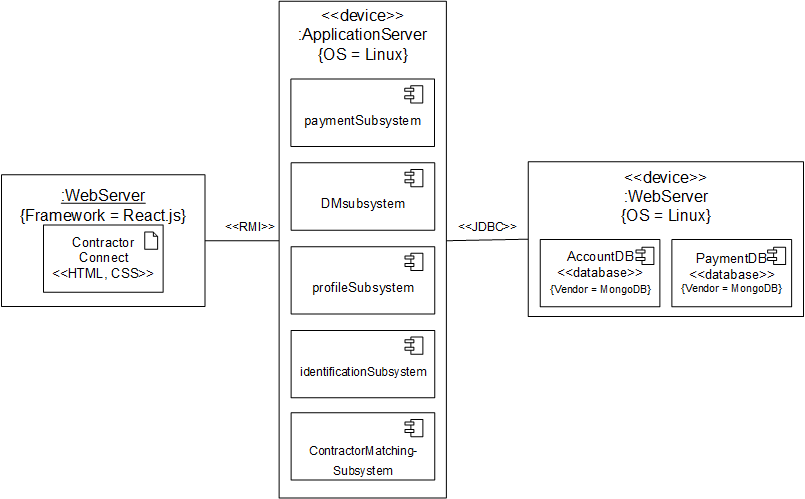
3.1 Pages – Amritpal



3.2 Home Page - Wilson

4

**4.1 Component/Deployment Diagram**



Gantt Chart

