Service Level Agreement

1. Agreement Overview

This Agreement defines a Service Level Agreement ("SLA" or "Agreement") between Coding Software Solutions (CSS) and ReMedi Healthcare Solutions (RHS) for the provision of information technology services necessary to support and maintain the web-based database application that CSS has developed for RHS. This SLA will remain in effect until it is superseded by a revised agreement that has been mutually endorsed by the parties involved. All IT (Information Technology) services covered by this Agreement are defined in terms that are understood by the primary stakeholders and are outlined in this SLA. Aside from anything explicitly stated herein, this SLA does not supersede existing processes and procedures.

2.Purpose, Objective & Goals

Purpose: Determine whether the necessary elements and commitments are in place for CSS to consistently provide information technology (IT) services RHS.

Objective: To provide RHS with a concise, coherent, and quantifiable description of the services we provide and make clear references to service responsibility, governance, roles, and/or responsibilities in the documentation. Match perceptions of expected service provision with actual service support and delivery.

Goals: To obtain mutual agreement for IT service provision between CSS and RHS.

3. Stakeholders

i.Service Provider: Coding Software Solutions ("Provider")

i.Client: ReMedi Healthcare Solutions ("Client")

4. Periodic Review

CSS will not be providing a periodic review for RHS as CSS are only providing a one-time IT service to RHS, which are development and deployment of the web-based database application only.

Development period: February 4th, 2022 – April 5th, 2022

Deployment on client's site date: June 1st, 2022

5. Service Management

CSS is responsible for the following specific service specifications for this SLA

Service scope for RHS:

- Business processes review (related to the web-based application development).
- Document review (related to the web-based application development).
- Development of the web-based database application for RHS
- Deployment of the web-based database application.
- Application testing and training.
- Future upkeep and maintenance (if needed)

6. Client (RHS) requirements

The following are responsibilities and/or requirements of RHS in support of this SLA:

i.All support such as, detailed reports, policies and other documents that would outline the business processes to support in the development of the web-based database application.

i.Availability of meetings, mentoring and guidance when required to aid and/or feedback in the development of the web-based database application.

7. Service Provide (CSS) requirements

The following are responsibilities and/or requirements of CSS in support of this SLA:

i.Meeting with RHS on a timely manner.

i. Keeping RHS up to date on the status of the web-based database application throughout the project.

i. Once completed, the web-based application will be deployed, and training will be conducted for RHS employees.

8. Service Assumptions

The following assumptions apply to in-scope services and/or components:

Any changes to services provided will be communicated to all stakeholders in a timely manner.

7. Service Management

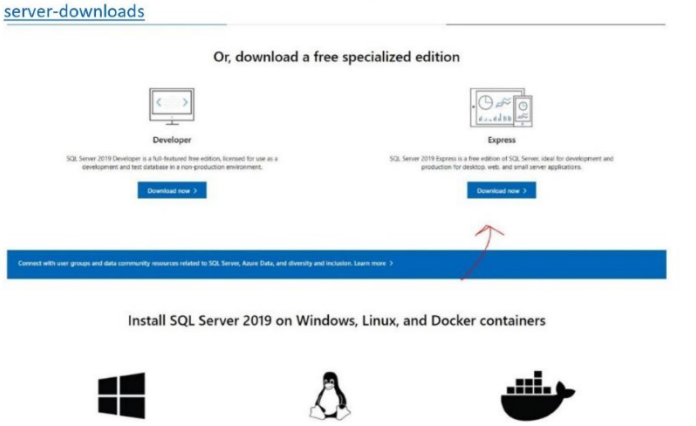
Consistent service levels provide effective support of in-scope services. The sections that follow give pertinent information on service availability, monitoring of in-scope services, and associated components.

Installation Procedures

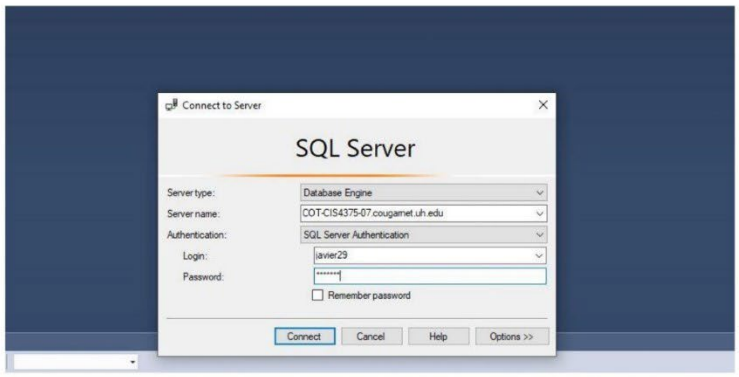
Installing and running the application on the local environment guide:

Note: these instructions are based on the assumption that it will be used and ran locally with a Microsoft SQL Server database

1. First, we need to make sure we have a database set up to hold our data. Like stated above we will use a Microsoft SQL Server Database
2. You can download a free version of it on here: https://www.microsoft.com/en-us/sql-server/sql-server-downloads

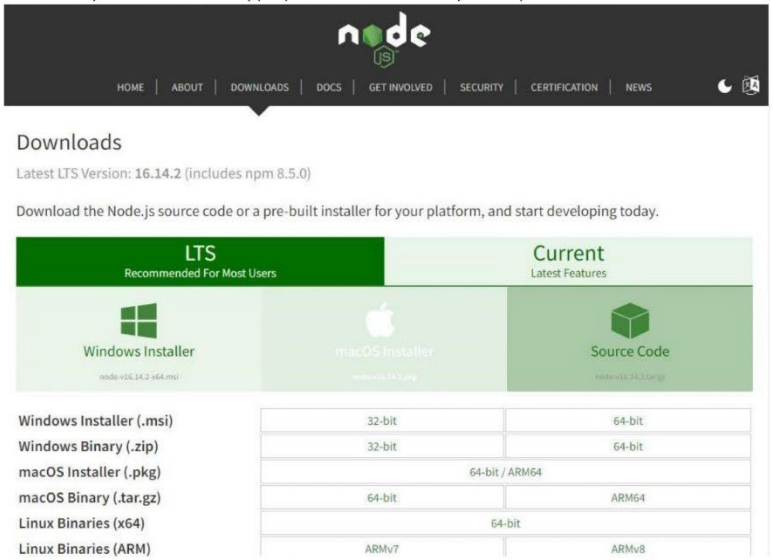


3. When installing make sure you opt to download Microsoft SQL Server Management as well. This will be the application you will use to modify and build your database
4. Once downloaded you will need to create a database and then run the scripts within the database provided to run and build all the tables you will need for the application
5. Make note of your login credentials as this will play a role later when connecting to the database through the backend



6. In order for your application to work you will first need to install both node.js and npm. To get this you can install them both from: https://nodejs.org/en/download/

7. Make sure you download the appropriate version LTS for your respective OS



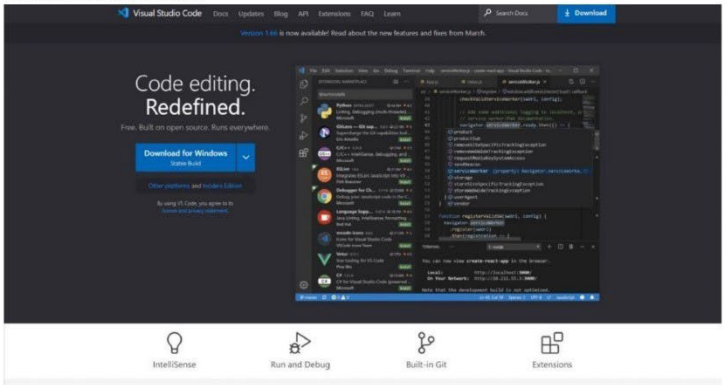
8. Once Installation is complete you will need to restart your computer
9. Open your terminal and type node -v to check and see if node.js install properly

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>node -v
v16.14.2

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>
```

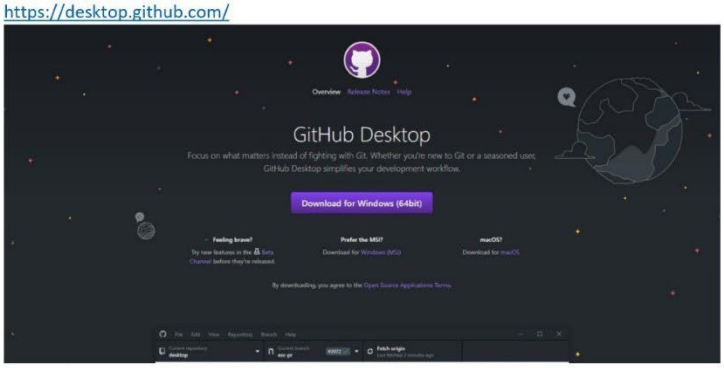
10. Type npm -v as well to make sure npm was installed

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>npm -v
8.5.0

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>
```

11. We will then need to download a couple of applications if it is not already downloaded. The first one is Visual Studio Code

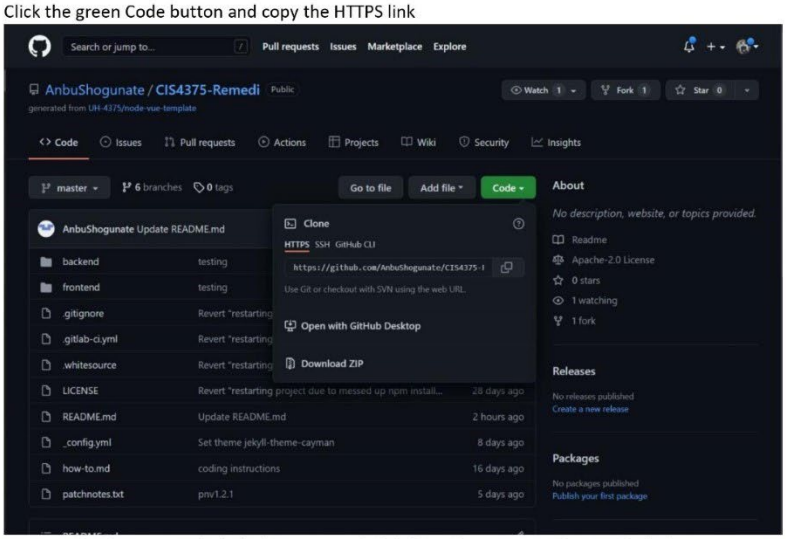


12. Then we will need to download an application called GitHub Desktop at: https://desktop.github.com/

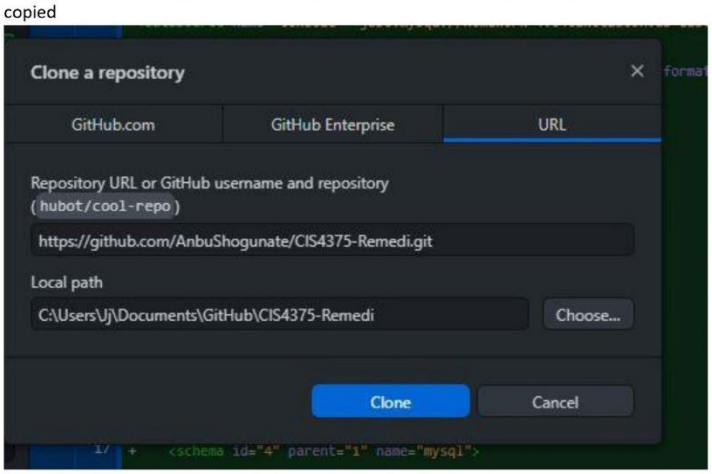


13. If you haven't created a GitHub account now would be the time to do so.
14. Once you have successfully created an account you can go ahead and navigate to the GitHub repo that contains the application at: https://github.com/AnbuShogunate/CIS4375-Remedi

15. Click the green Code button and copy the HTTPS link

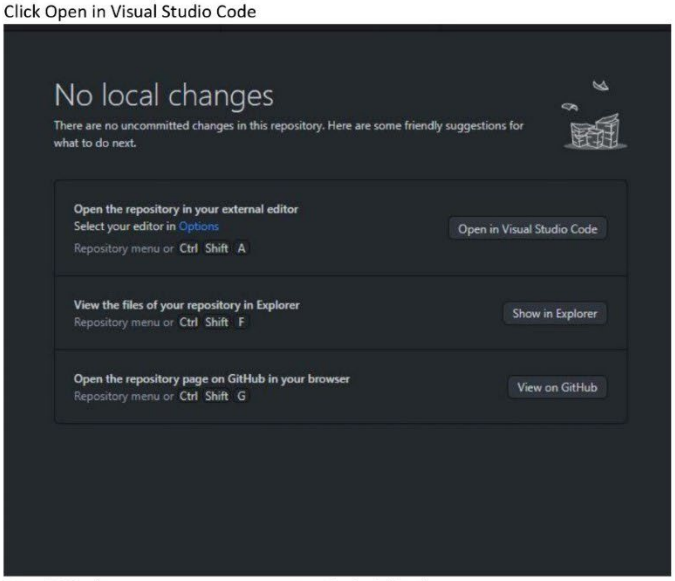


16. Navigate back to your github desktop app and click file> clone repo and paste the link you copied

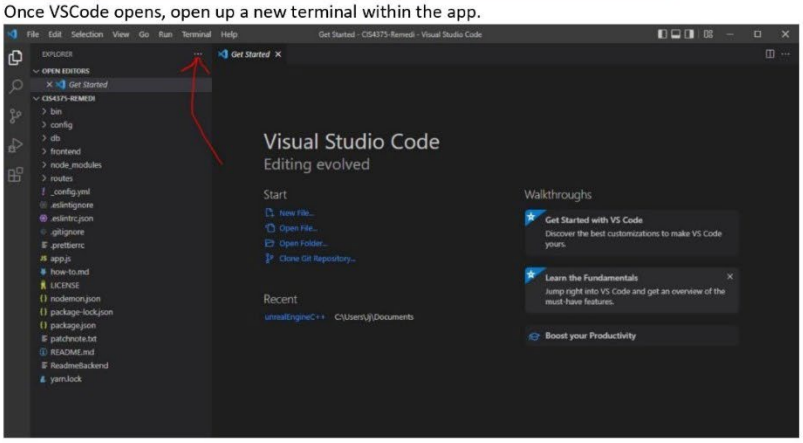


17. Switch to the appropriate branch if needed

18. Click Open in Visual Studio Code



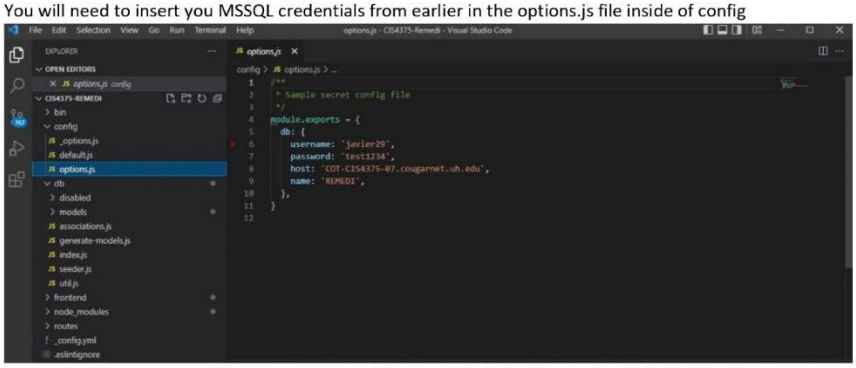
19. Once VSCode opens, open up a new terminal within the app.



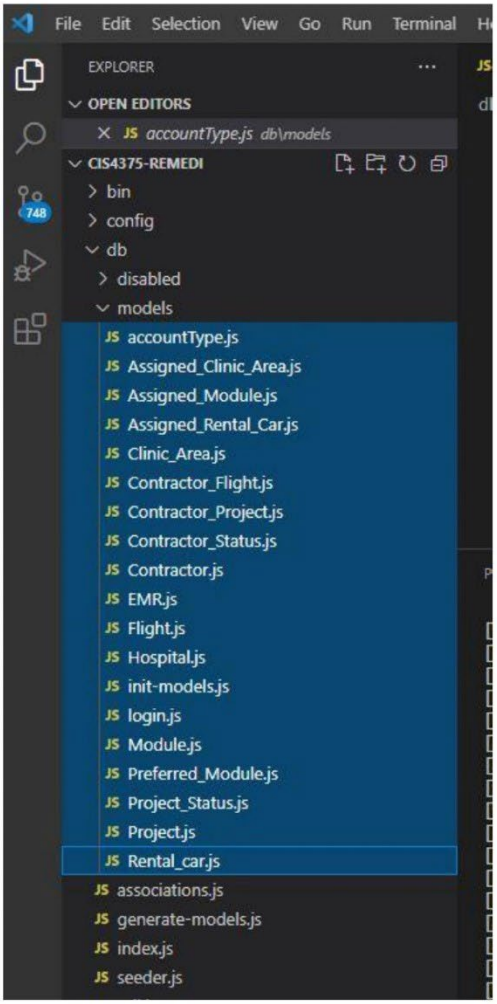
20. In the terminal type in npm install

```
C:\Users\Jj\Documents\GitHub\CIS4375-Remedi>npm install
npm WARN deprecated uuid@3.3.2: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstanc
es, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API
 surface has changed to use Promises in 1.x.)

changed 28 packages, and audited 650 packages in 5s

5 packages are looking for funding
  run `npm fund` for details

21 vulnerabilities (6 moderate, 11 high, 4 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 8.5.0 -> 8.6.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.6.0
npm notice Run npm install -g npm@8.6.0 to update!
npm notice

C:\Users\Jj\Documents\GitHub\CIS4375-Remedi>
```

21. Type cd frontend and in here run npm install as well
22. Once that is completed you will type cd .. to go back to the root directory
23. You will need to insert you MSSQL credentials from earlier in the options.js file inside of config

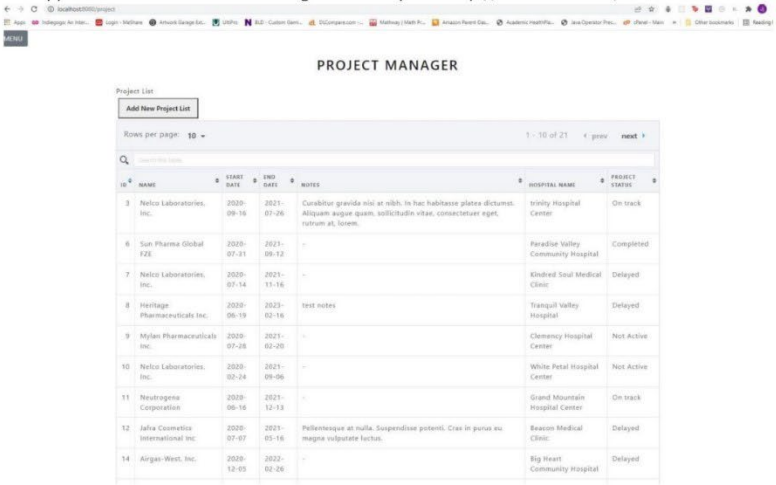


24. In the root directory you will run npm run startAll to start both the frontend and the backend together * A sequelize error might occur , if this is the case you will need to follow a few additional steps to get the application running*
25. If the error above happens you will need to delete the models



26. Then in the terminal type npm run generate-models in the root directory to re-generate
27. Run npm run startAll *note: if this command doesn't work you will need to run npm run start on both the frontend and backend directory*

28. The application should now be running successfully on : http://localhost:8080/



ID	NAME	START DATE	END DATE	NOTES	HOSPITAL NAME	PROJECT STATUS
3	Nelco Laboratories, Inc.	2020-09-16	2021-07-26	Curabitur gravida nisi at nibh. In hac habitasse platea dictumst. Aliquam augue quam, sollicitudin vitae, consectetuer eget, rutrum at, lorem.	trinity Hospital Center	On track
6	Sun Pharma Global FZE	2020-07-31	2021-09-12	-	Paradise Valley Community Hospital	Completed
7	Nelco Laboratories, Inc.	2020-07-14	2021-11-16	-	Kindred Soul Medical Clinic	Delayed
8	Heritage Pharmaceuticals Inc.	2020-06-19	2023-02-16	test notes	Tranquil Valley Hospital	Delayed
9	Mylan Pharmaceuticals Inc.	2020-07-28	2021-02-20	-	Clemency Hospital Center	Not Active
10	Nelco Laboratories, Inc.	2020-02-24	2021-09-06	-	White Petal Hospital Center	Not Active
11	Neutrogena Corporation	2020-06-16	2021-12-13	-	Grand Mountain Hospital Center	On track
12	Jafra Cosmetics International Inc	2020-07-07	2021-05-16	Pellentesque at nulla. Suspendisse potenti. Cras in purus eu magna vulputate luctus.	Beacon Medical Clinic	Delayed
14	Airgas-West, Inc.	2020-12-05	2022-02-26	-	Big Heart Community Hospital	Delayed

Project Maintenance

Our solution for ReMedi uses Microsoft SQL Server. Microsoft has detailed documentation about how to perform the following maintenance activities for the database. The following links will be labeled so that ReMedi can use the information to perform maintenance.

Data Backup

Microsoft has a detailed page for data backup (with instructions, examples of a script and a screenshot) in the following link:

https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/create-a-full-database-backup-sql-server?view=sql-server-ver15

Data Cleanup

Microsoft has a detailed documentation page for data cleanup in the following link:
https://docs.microsoft.com/en-us/sql/data-quality-services/data-cleansing?view=sql-server-ver15

Data Recovery

Microsoft provided a detailed documentation page on how to restore and recover a database based on what type of recovery (entire database, data file, or data page) in the following link:
https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-and-recovery-overview-sql-server?view=sql-server-ver15

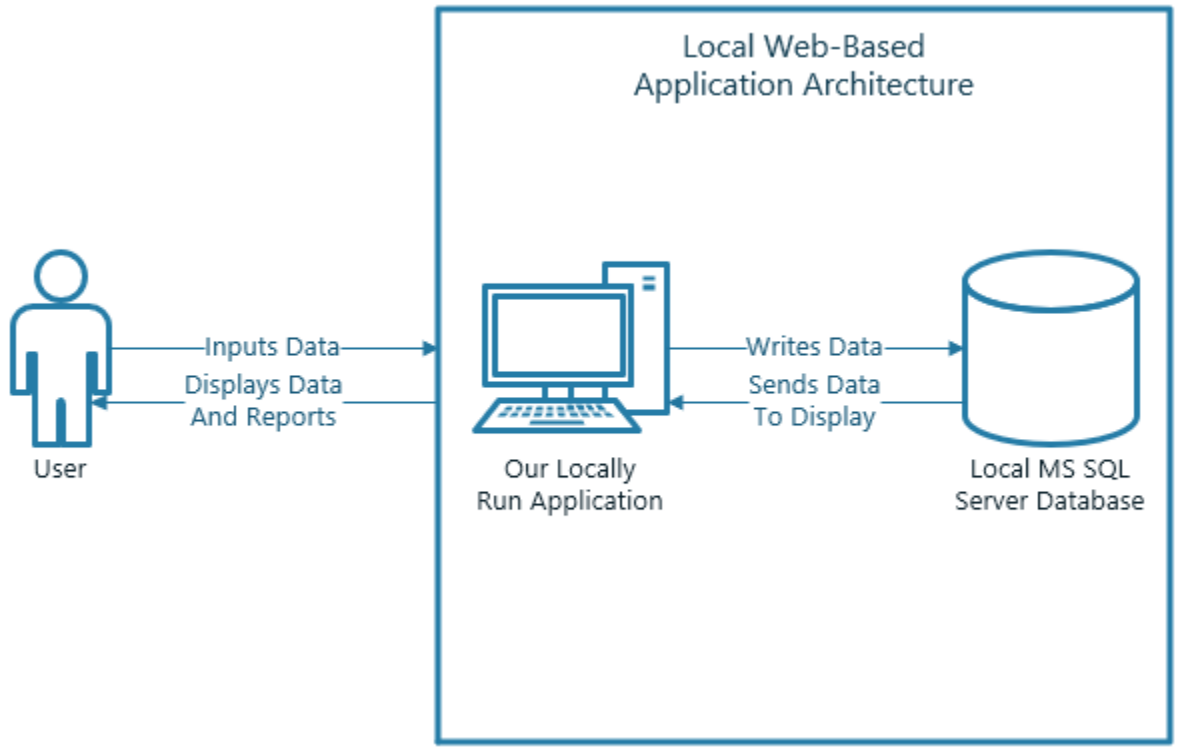
Data Archival

There is no official documentation about archiving data on the Microsoft SQL Server site, so we will list the steps on how to archive data. These steps work under the assumption that the data should still be accessible and that it will be saved in the same database.

1. Select a table you want to archive (we will use Contractor as an example)
1. Create a new script
1. The script will follow this language
    a. SELECT * INTO ArchiveContractor
    a. FROM Contractor
1. That creates a new archive table with the name ArchiveContractor

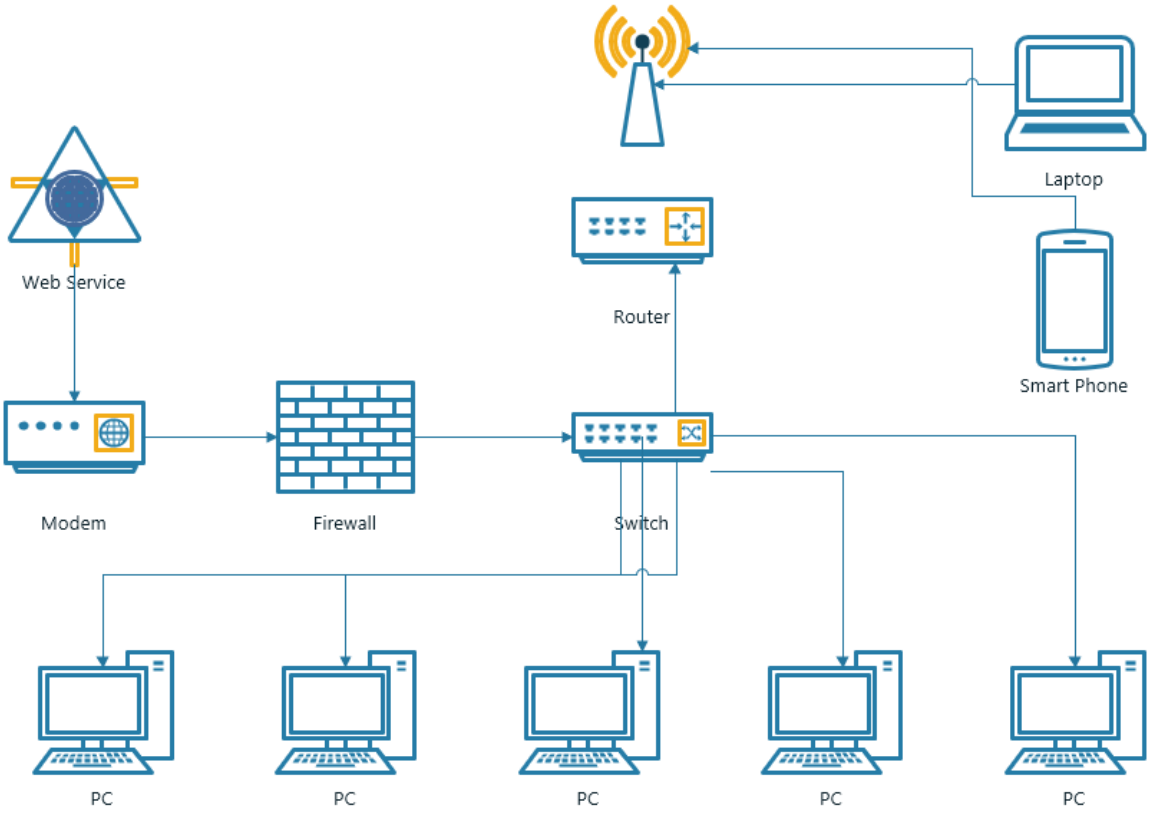
These steps would be repeated until all tables that ReMedi wants to archive are archived.

Application Architecture



Page Break

Network Architecture



Page Break

Organization's Policies

Atilio Molina met with ReMedi on Monday April 11th to get an understanding about their organization's policies. When they were asked, Carola (our sponsor) and GP (the other employee who has been on some of the meetings) said that they were still creating some policies. Now, ReMedi does have a completed document or set of policies for the following sections.

Security Policy

At this moment Remedi Health Solutions does not have a policy for Security.

Outsourcing Contract Policy

At this moment Remedi Health Solutions does not have a policy for Outsourcing Contracts.

Customer Relationship Policy

The customer relationship policy is currently in development by Remedi Health Solutions.

Disaster Recovery Policy

The Disaster Recovery policy is currently in development by Remedi Health Solutions.

Page Break

Project Future Improvements

Identifier	Future Improvement	Description	Priority
1	Create a Project Menu Page	This would be a page that would list all the projects currently being worked on as clickable buttons.	3

		When a project is clicked, it would take the user to a page that shows them information about the project, the hospital it is associated with, and a list of contractors currently assigned to that project.	
2	Cloud Deployment	ReMedi should deploy our application to be hosted on a cloud server. This way, the application is accessible to anybody and is not constrained to one or two computers.	2
3	Push Notifications	This would send directors notifications in their emails whenever changes to a contractor or a project were made by anybody.	4
4	Attach resumes to Contractors	A function that would let the hiring director attach a contractor's resume to their information page for easy access	5
5	Contractor Access	The application would let contractors outside of the ReMedi network access the website. They would have read-only access and only be able to see information related to them and the projects they are assigned to. They could also see any travel information they may have	6
6	Calendar Integration	The dates for flights, projects, and rental cars would be pushed to the director's calendar apps. This way they can view important dates on their calendars without having	7

		to access the app for that information	
7	Login Security Clearence	An update to add security clearances to individual logins. This would grant specific CRUD access to users depending on their position in the company.	1

Page Break

Application Development Lessons Learned

What Worked

Throughout our development process, various steps are used to complete our project. First, using a pre-built template that our lead developer previously worked on saved us a lot of work. It allowed us to have a concrete framework that we knew already worked. Second, using Vue and Node.js, were languages that many of us have used in the past, which allowed us to skip learning a new language. Third, using MSSQL, allowed us to visualize the data easily without going through Postman or console.log. Fourth, having all members use Discord over GroupMe or Teams. Using Discord creates a 24-hour communication cycle that was very helpful for our development team. For instance, if an issue occurs when connecting the backend to the database, it would be posted and quickly remedied within the hour. Fifth, separating GitHub repos for different parts of the application development. It's to prevent codes from being worked on top of each other and causing problems when pushing and pulling from different branches. And lastly, a critical benefit to us was dividing the code evenly among the development team. It not only creates fairness but also reduces workload.

What did not Work

One of the earliest work habits we experienced to be detrimental was coding simultaneously on the same section. It created issues when merging repos and set us back on development. The second hurled was not using a template made for cloud deployment. If dependencies depreciate during production, some features may not be available. In this case, cloud deployment. Third and lastly, coding before finalizing the database. It caused us to get bottled necked, have to remake code, and constantly make updates to our code.

What Could Be Done Differently

One process that we could do differently is to try and deploy the application earlier in the development timeline. But with our time management, the window to deploy was too small. Another thing we wished had done differently was to finalize our full-stack earlier. This falls into another thing we wish we had done differently. That is if we have more time for workshops for the whole project team early in the stages. So, in general, the biggest constraint was time for us.

Page Break

New Technologies Learned and Utilized

For this application we used a variety of different technologies and frameworks. The main stack of the application was built using Vue.js as the front-end framework and utilizing node.js with express on the backend. For the database we used Microsoft SQL Server. Our application was developed and ran by Visual Studio Code. Some of the new frameworks and technologies that we implemented on the front-end were:

Front-End

- Bootstrap Vue-used for styling and form creation.

- Bootswatch-more styling libraries
- VueFormulate -helped with form validations
- Vue router- Used for page navigation
- Axios- Used to help make RESTful API requests to the backend
- Vue-Good-Table- Helped in creating the tables needed to display a variety of data
- SweetAlert2- Implemented to display pop out messages such as errors and confirmations

For the backend we used technologies and libraries provided by node.js/express. Some other new technologies used and implemented were:

Backend

- Sequelize- Used for modeling our database
- Express router- Enabled us to be able to reach different endpoints
- Cross-Origin Resource Sharing (CORS)- Allows permissions from outside connections to make HTTPS request
- Concurrently- Implemented to provide a single command to concurrently run frontend and backend together

Page Break

References and Citations

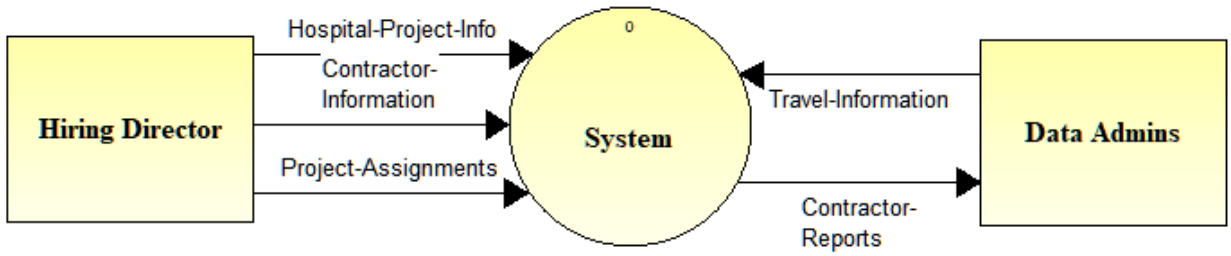
Stack Overflow
Template for UI:
https://github.com/mubaidr/node-vue-template
Previous 3365 Code (only Weizhao's code was used):
https://github.com/AnbuShogunate/uh_projects_3365

Page Break

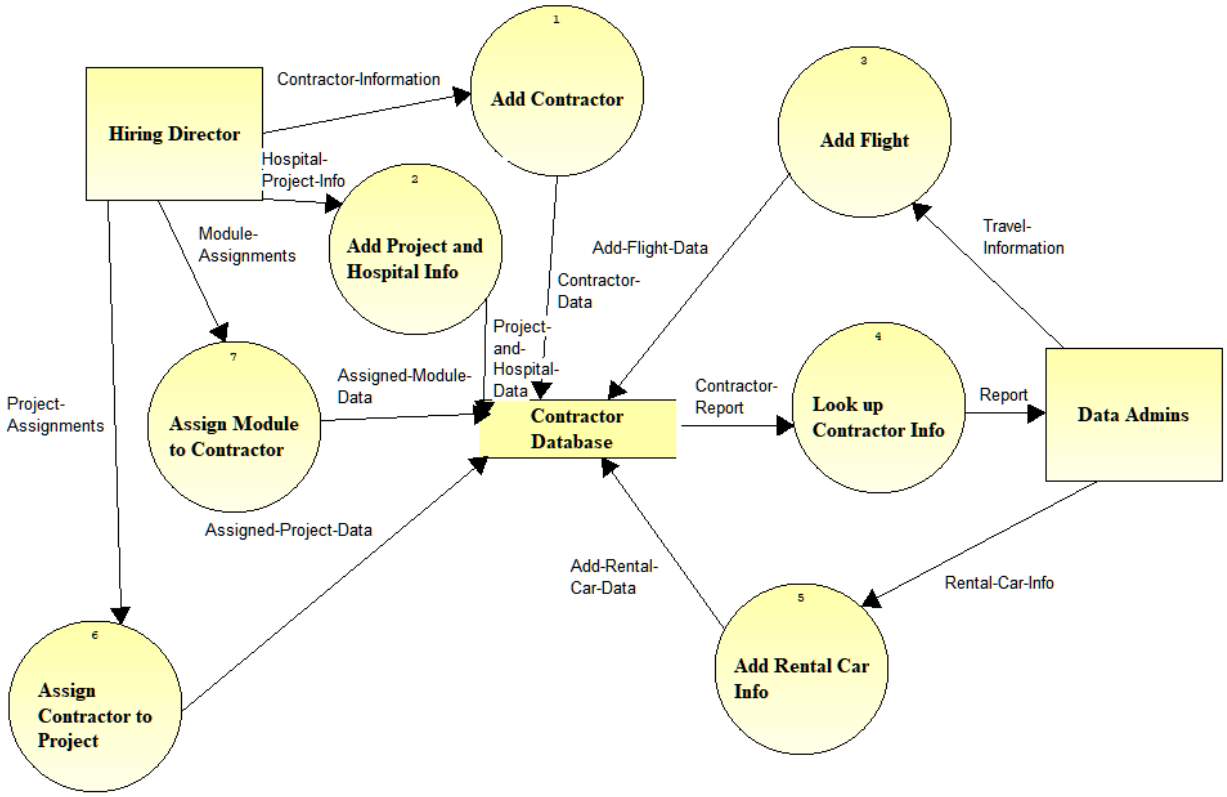
Appendix

Data Flow Diagrams

Level 0 Diagram

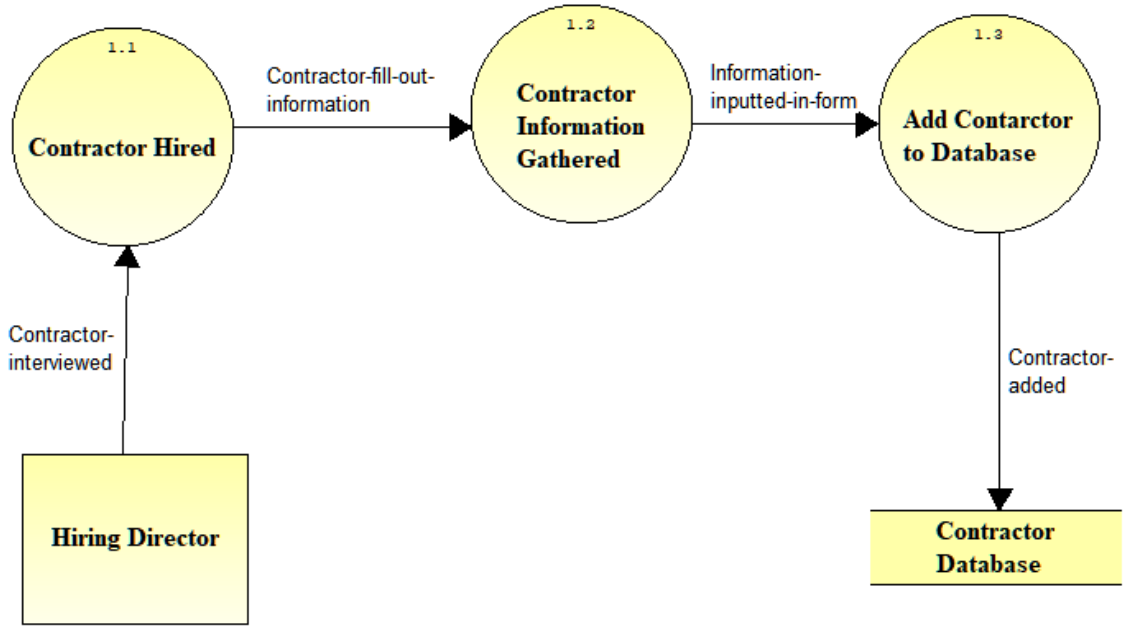


Level 1 Diagram

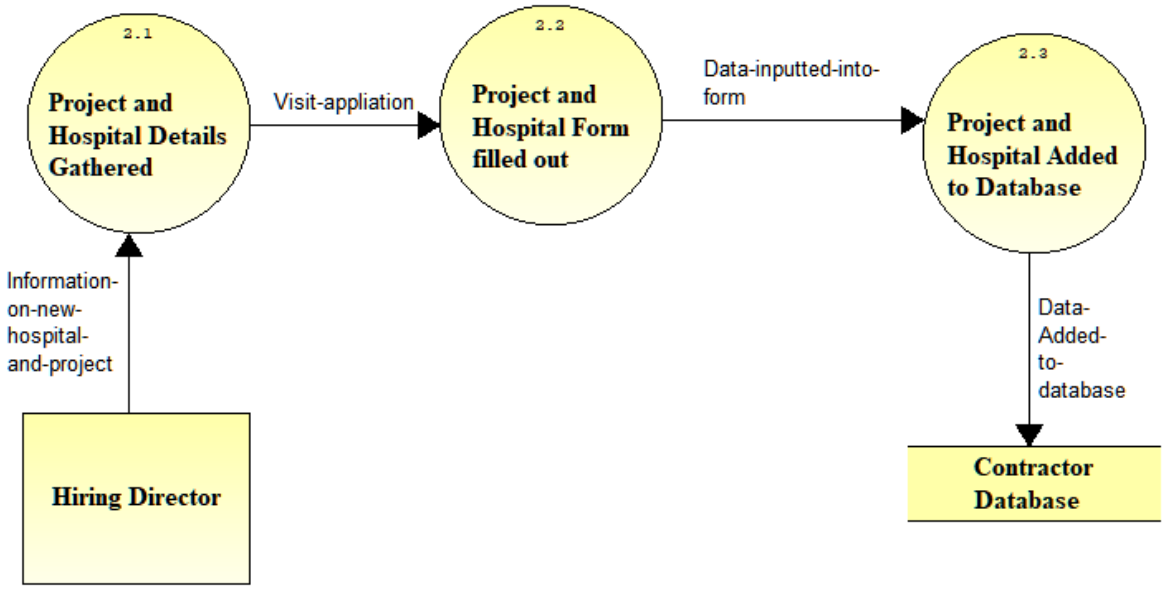


Page BreakLevel 2 Diagrams

Process 1

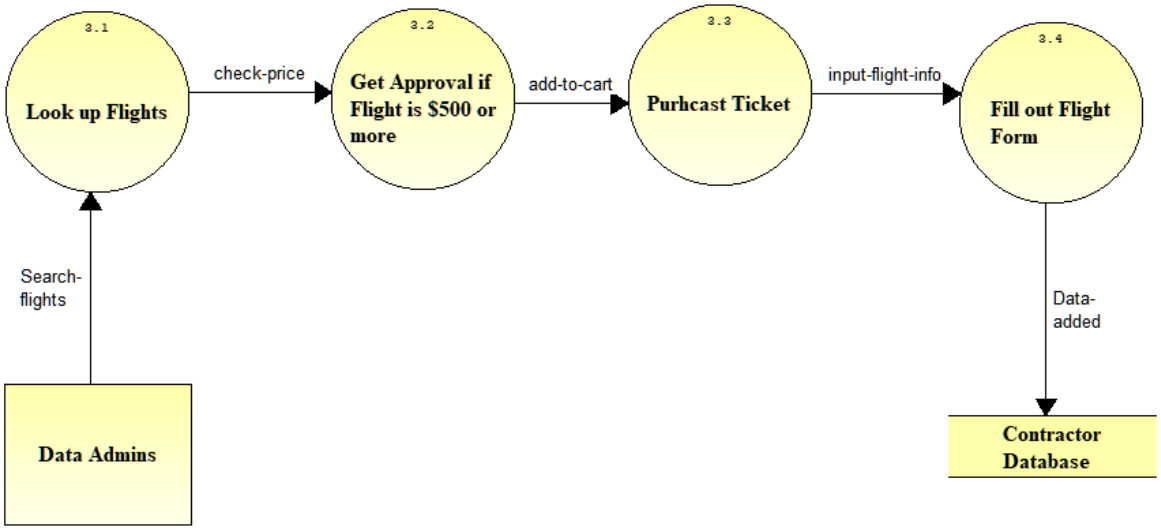


Process 2

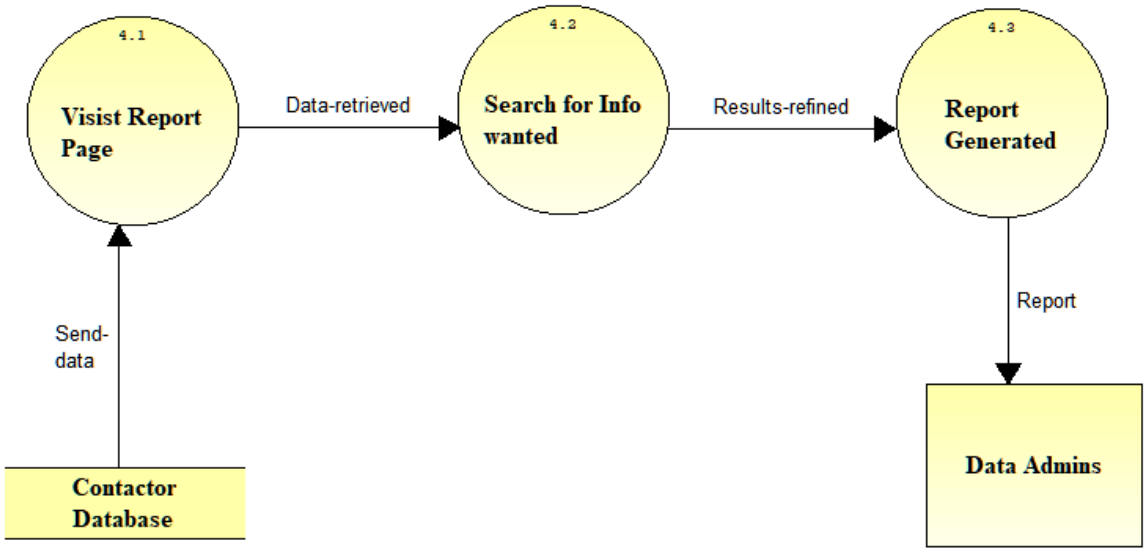


Page Break

Process 3

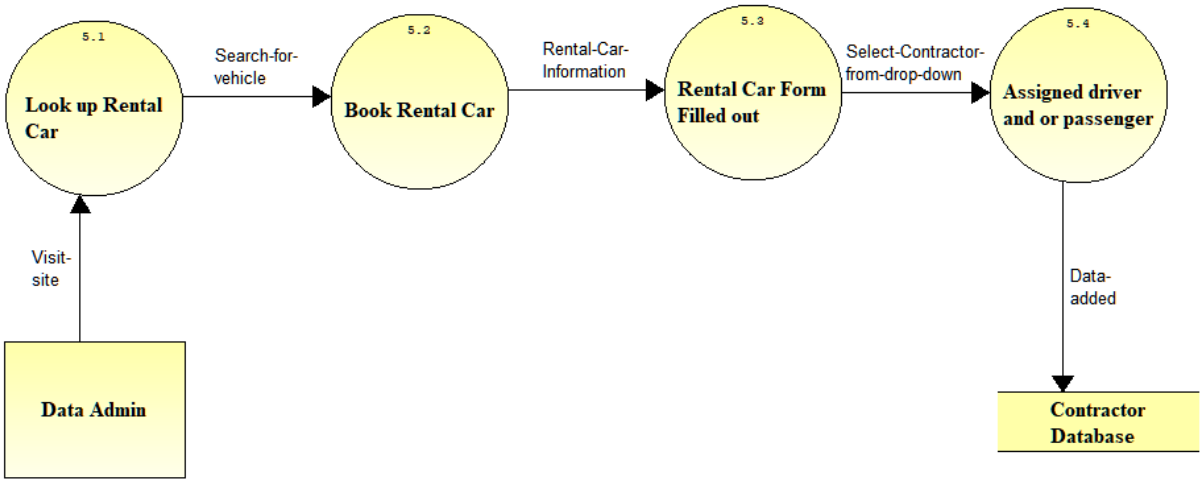


Process 4

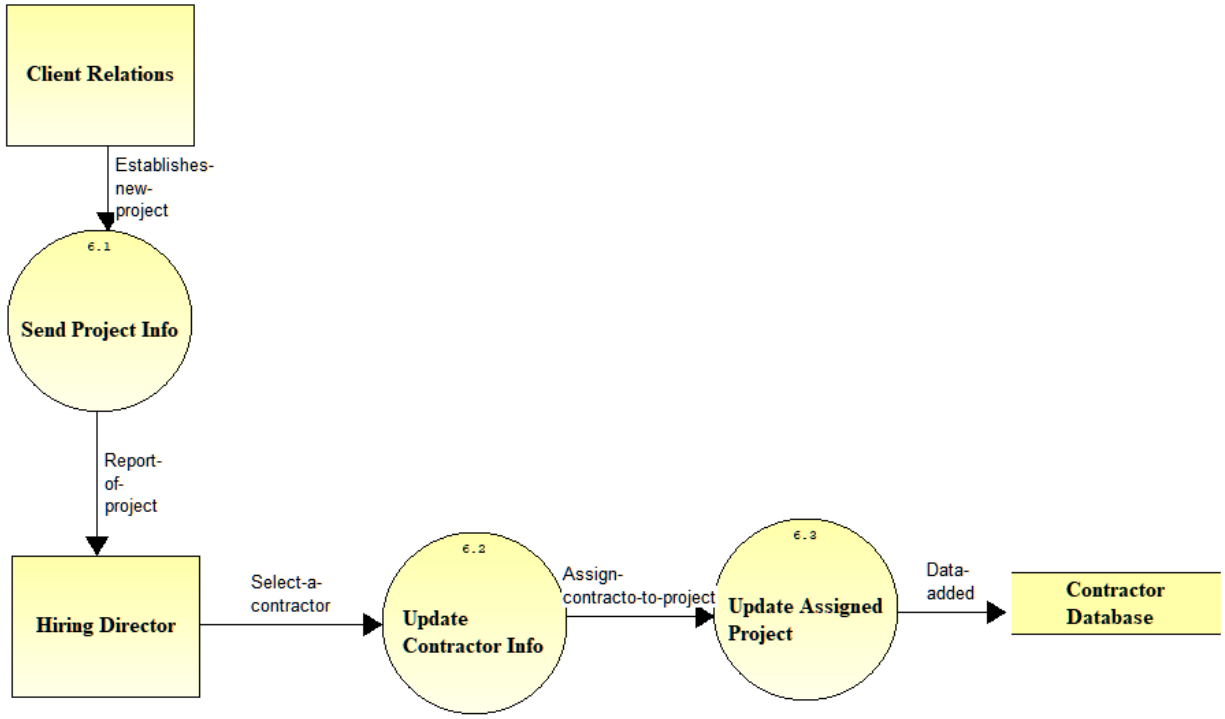


Page Break

Process 5

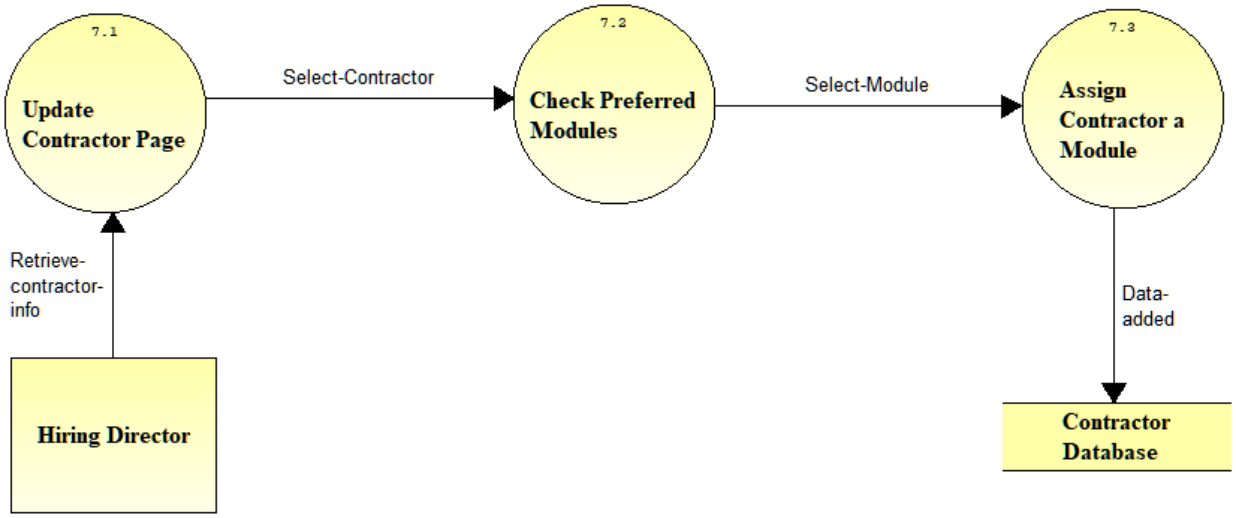


Process 6



Page Break

Process 7



Page Break

Use Cases and Activity Diagrams

Use Case Name: Add Project	
Actor(s):	Salesperson, Client, Data Admin
Description:	Add new projects into the system

Triggering Event:	An employee at Remedi adds a project into the application
Trigger Type:	[ ] Internal [X] External
Steps Performed:	1. Remedi salesperson meets with potential Client 1. Client signs contract agreeing with project details and rates for contractors 1. Contract details are sent to spreadsheet employee 1. Spreadsheet employee visits the web-application 1. Spreadsheet employee inputs project details 1. Project is added successfully to the database 1. Project details can be seen in the web-application
Preconditions:	1. Remedi has a signed contract with the Client 1. Contract details have been sent to the Remedi employee
Postconditions:	Successful: 1. Project and project details are added to the application Unsuccessful: 1. Project is unable to be added to the application because it already exists or application error
Assumptions:	Salesperson is sending the correct contract to the data admin to input into the database. The project has not been added to the database. The application will function correctly when someone adds a project.
Requirements Met:	Project/Client does not already exist.
Outstanding Issues:	None
Priority:	High
Risk:	Low