Career Finder System

Website Application

Group Number 11

Team Members

Ahmed Elselmy **6449**

Nada Samy **6357**

Mariam Elraggal **6429**

Table of contents:

Requirements:

User Requirements:

Identifier	Requirement
REQ1	The system shall allow users to log in into the system if they already have an account.
REQ2	The system shall allow new users to register an account.
REQ3	Any job seeker who logs into the system shall be able to view all jobs posted and apply to jobs posted in the jobs feed.
REQ4	Any company admin who logs into the system shall be able to add job posts or remove job posts and also be able to view all the applications on a certain job.

System Requirements:

Identifier	Requirement
REQ1	A database is needed to store all the needed information.
REQ2	Show login form to users and validate their login info with the database.
REQ3	Show registration form to users when the register button is pressed and authenticate their information before successful registration.
REQ4	The system shall allow a job seeker to see all companies posted jobs details like job title, name of company who post the job, job requirements, job description, job category, and deadline of application.
REQ5	Show application form to job seekers when the apply button is pressed on any job post.
REQ6	Validate all the input data job seeker enters in the application form.
REQ7	The system shall show a form to the company admin to fill the details of the new job post when the create post button is pressed.
REQ8	The system shall validate the data of new job post and add new job post information in the database when the company admin creates a new post.
REQ9	The system shall allow to the company admin to remove any job post.

Functional Requirements:

Identifier	Requirement
REQ1	Construct a database schema and tables.
REQ2	When a user tries to log into the system, check his info with the database.
REQ3	If the login info is not correct, show error message and allow user to try to login again.
REQ4	Each user is identified with his unique email that is stored in the database.
REQ5	When a user tries to register a new account, update the database with the user's info after validating the information.
REQ6	Validate the input data in the registration form by checking unique email format, minimum password length equal to 5 and maximum 12, maximum name for user equal to 30, and password and confirm password must be the same.
REQ7	If the registration information is not correct/already exists, show error message and allow user to try to register again.
REQ8	If the registration process success, the website will redirect to the login page with success message.
REQ9	Validate a job seeker's application when the submit button is pressed.
REQ10	Update the database with a user's application to a job when his application is submitted.
REQ11	Allow company admin to create new job posts if it does not already exist in the database.
REQ12	Add the new job post information to database after validation.
REQ13	Show list of all applications on a job to the company admin.
REQ14	End user session when the logout button is pressed.

Non-Functional Requirements:

Identifier	Requirement
REQ1	The system shall be responsive, user friendly.
REQ2	The system shall not show any time lag when performing requests.
REQ3	The system shall not fail to update the database when needed.

Software Process

Suggested type of software process: Agile

Division of phases:

Phase 1) Model design & validate data with user.

This phase includes implementing:

- Database schema migrations design
- Implement database model
- Database admin seeder

Phase 2) Website home page UI & Login and register UI and logic.

This phase includes implementing:

- Website homepage UI view
- Login page UI view
- Job seeker's register page UI view
- Login function logic
- Job Seeker 's register function logic
- Logout function logic
- Middleware for protecting routes
- Testing

Phase 3) Website admin portal page UI design & logic implementation.

This phase includes implementing:

- Admin dashboard UI view
- Add company page UI view
- Handle statistics that will be shown to admin
- Add company function logic
- Delete company function logic
- Testing

Phase 4) Company portal page UI design & logic implementation.

This phase includes implementing:

- Company dashboard page UI view
- Add job post page UI view
- Add job post function logic
- Applications page UI view for specific job chosen
- Applications function logic get data from database
- Delete Job post function logic
- Testing

Phase 5) User portal page UI design & logic implementation

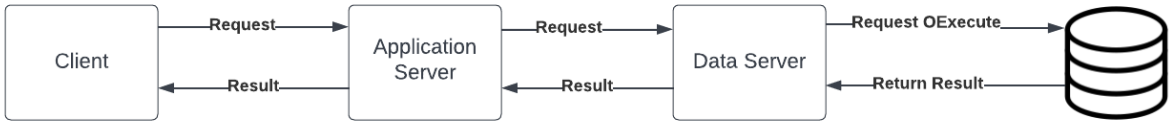
This phase includes implementing:

- Jobs' feed page UI view
- Job's feed function logic
- Apply to job page UI view
- Apply to job function logic
- Testing

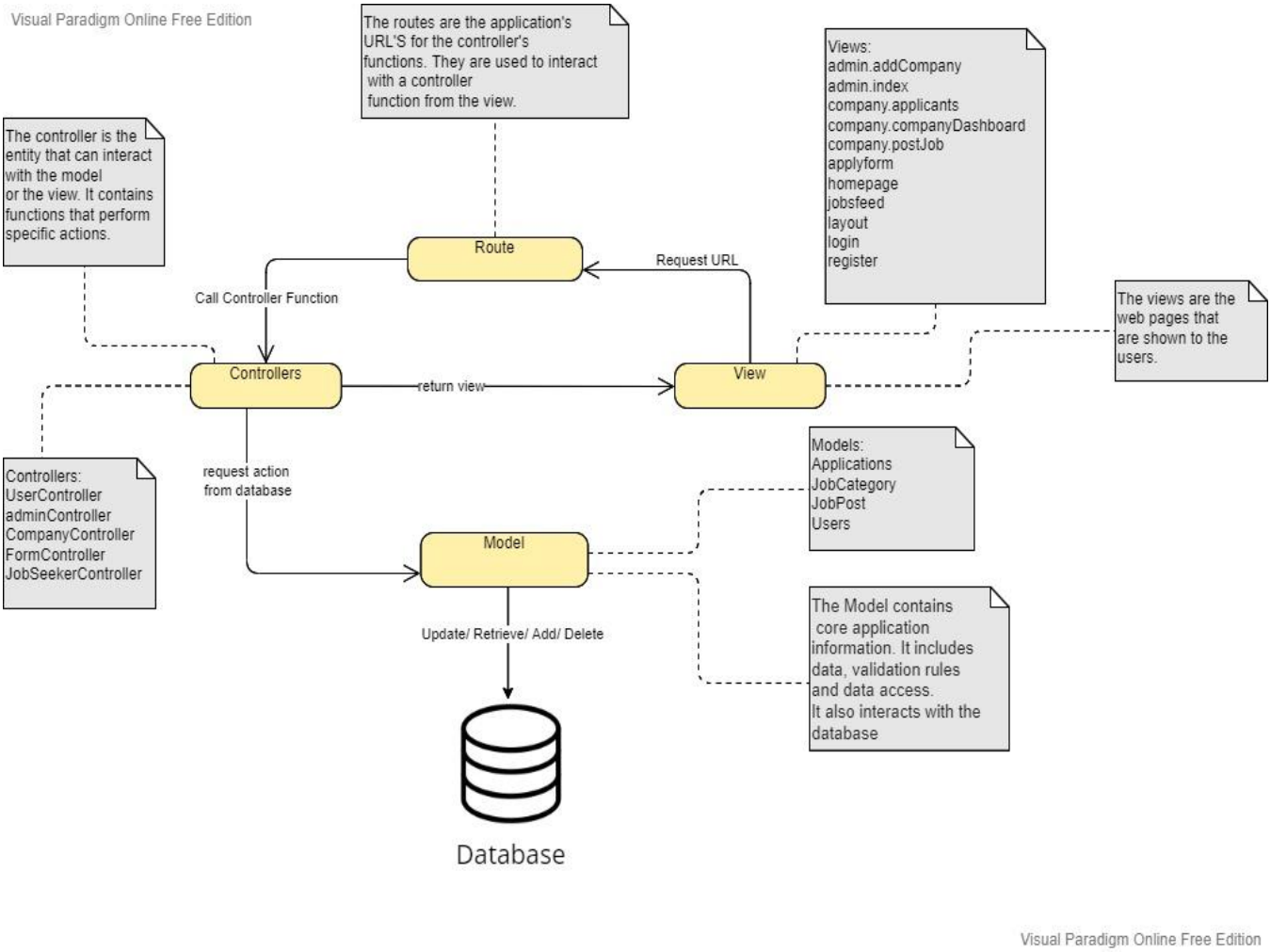
Architectural Design:

System Architecture:

A three-tier client/server is a type of multi-tier computing architecture in which an entire application is distributed across three different computing layers or tiers. It divides the presentation, application logic and data processing layers across client and server devices.

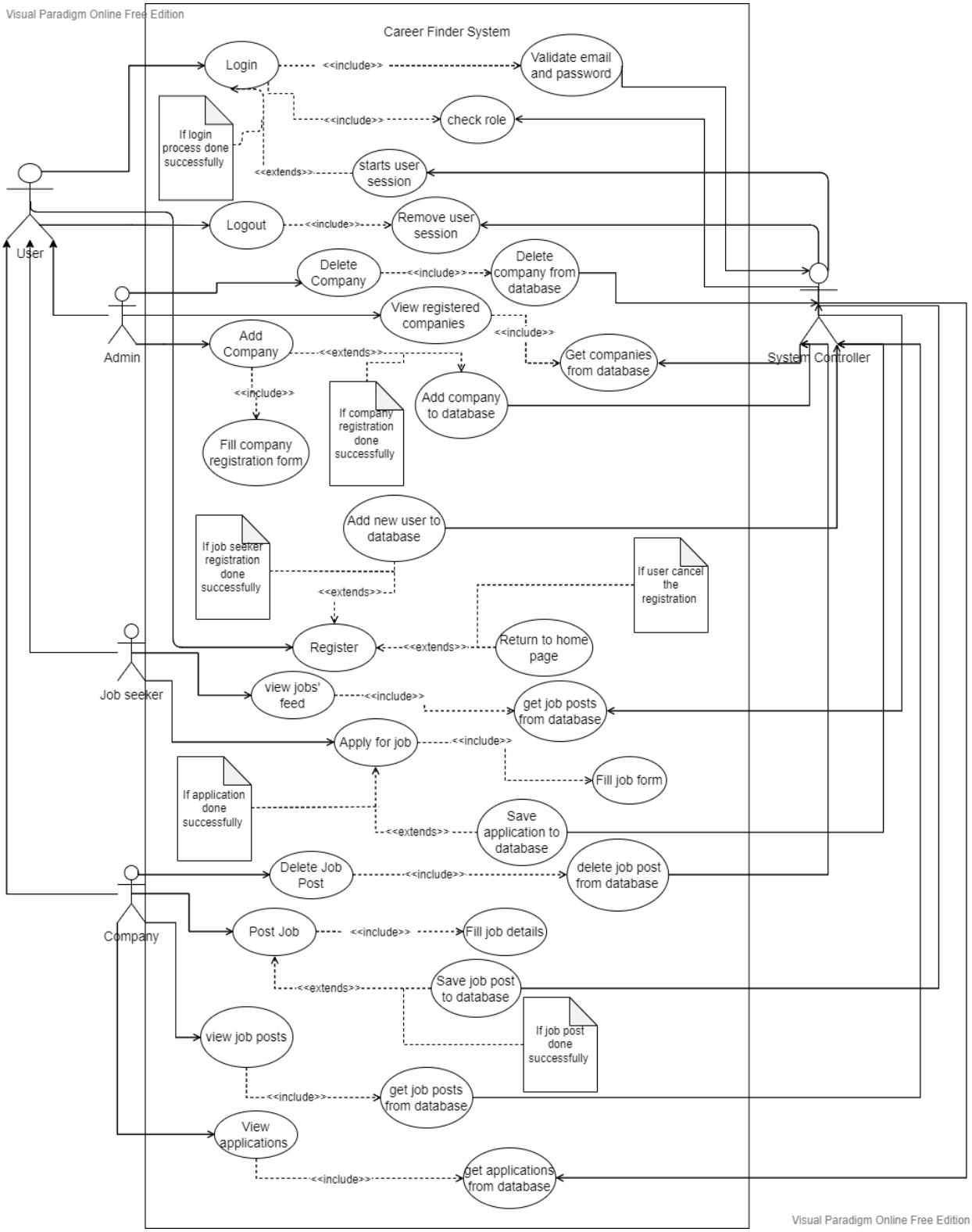


Web application MVC architecture:

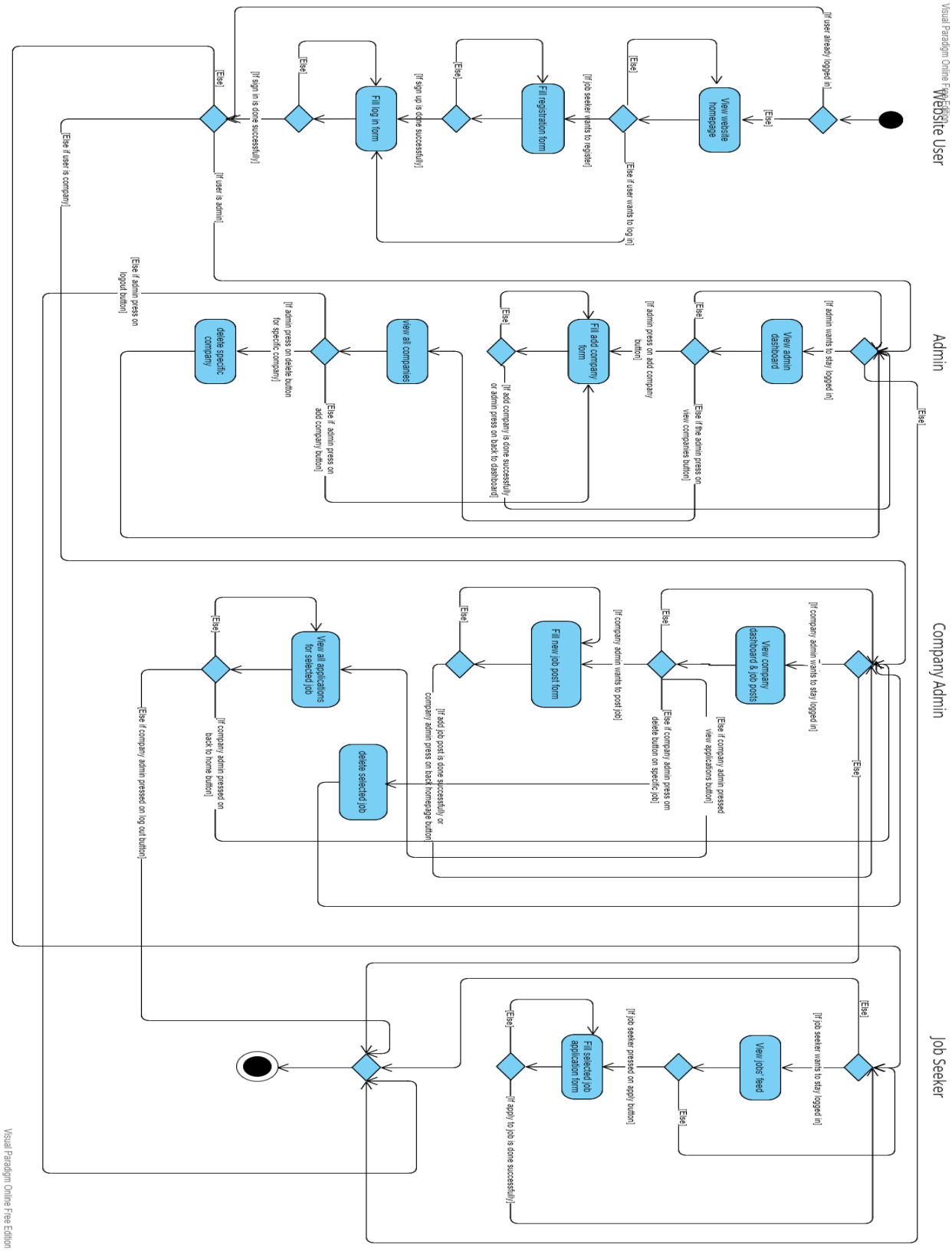


UML Diagrams

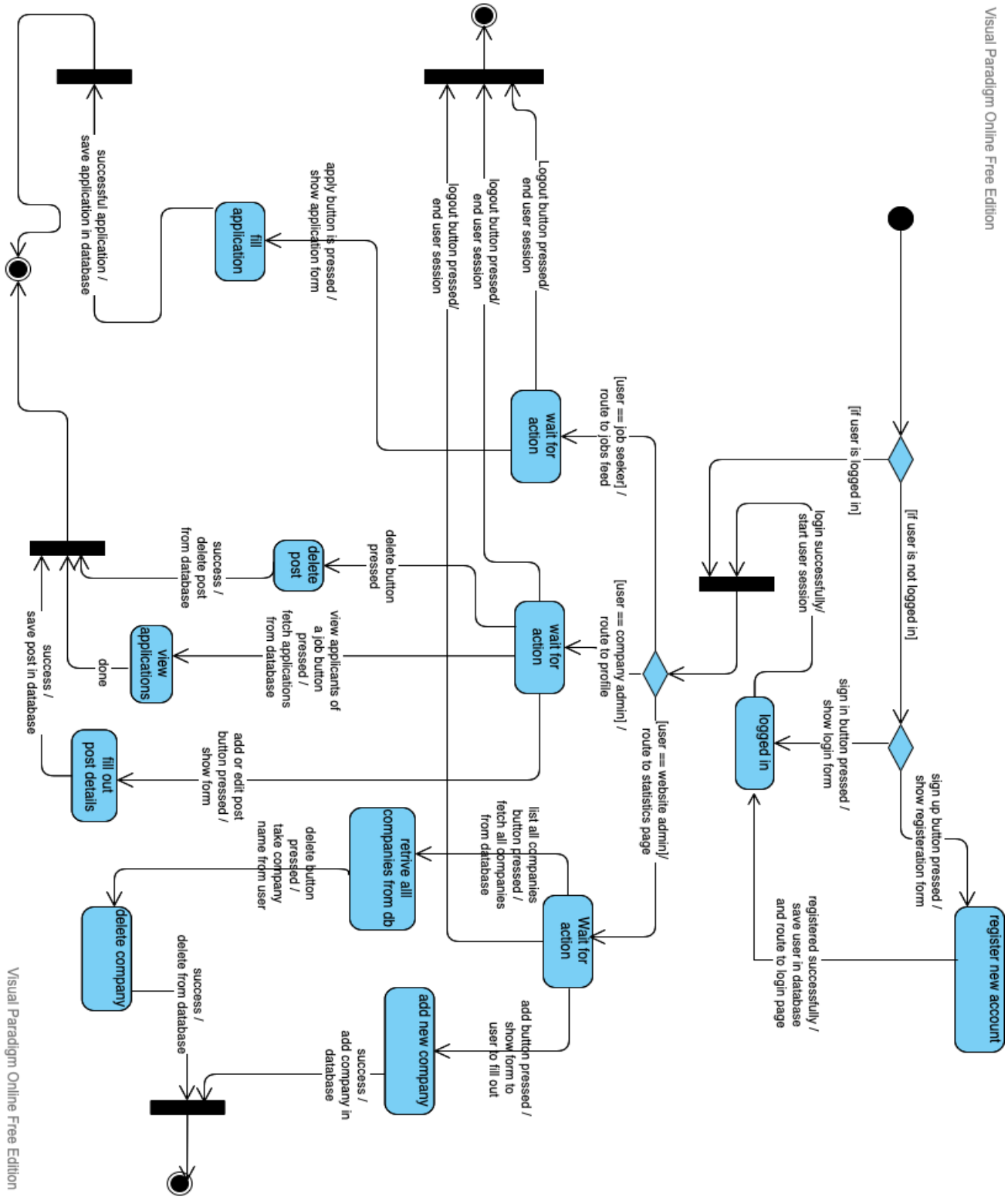
Use Case:



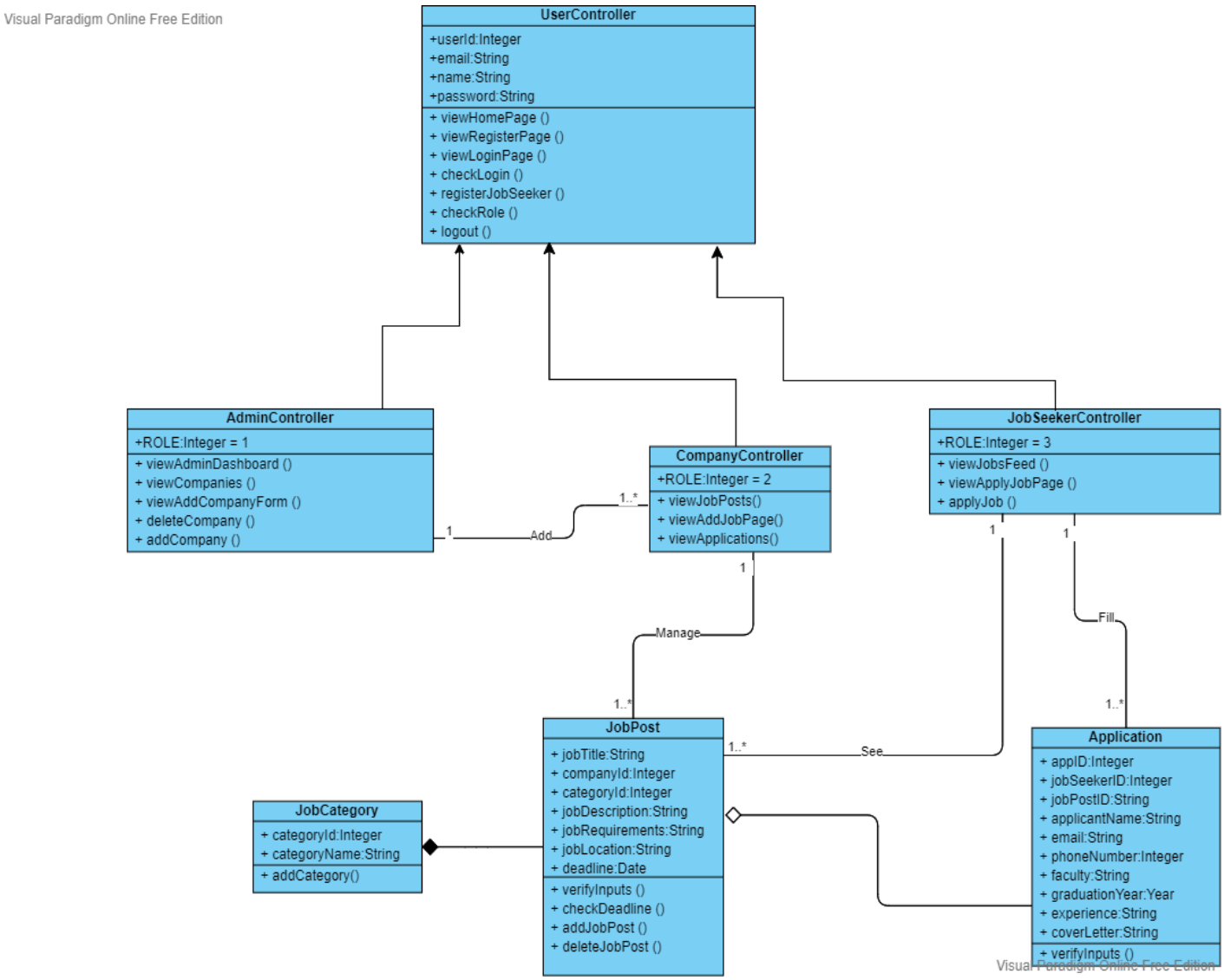
Activity Diagram:



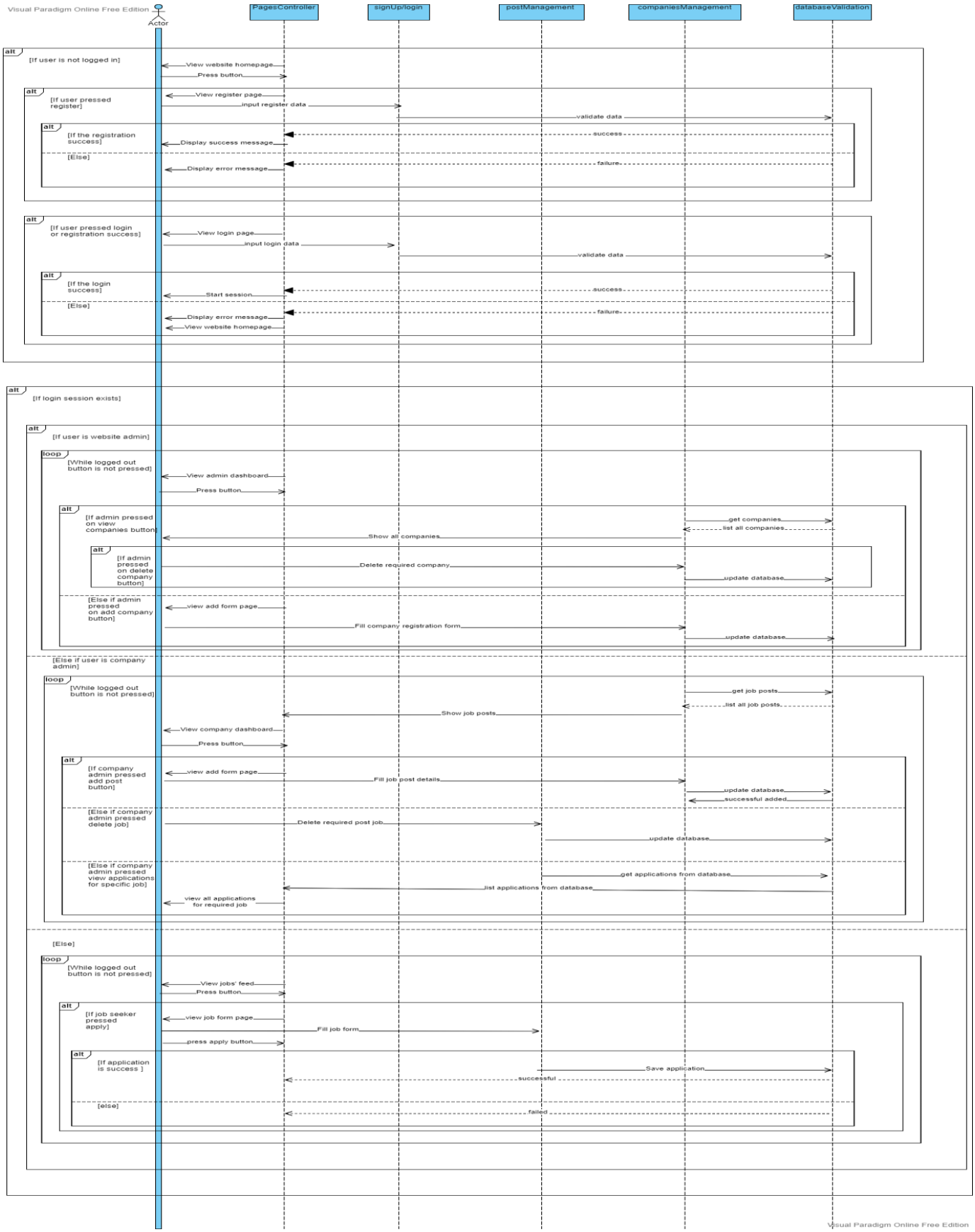
State Diagram:



Class diagram:



Sequence Diagram:



Use case diagram description:

There is the User actor who displays common functionality between the three types of users who use the system (Job Seeker, Company Admin, and Website Admin) which is to login, register or logout. The Job seeker, website admin, company admin actors inherit this from the user actor.

The system controller is involved in checking the data before updating the database whether adding new data to the database or deleting data from the database. The system controller also retrieves data from the database.

Activity diagram description:

This diagram shows the steps of actions that any user of the system can make once the website is opened whether that user is a job seeker, company admin or website admin.

State machine diagram description:

This diagram shows the various states of the system and how the system switches from one state to another depending on the actions the system controller receives from various users. Most states include the system controller retrieving/adding/deleting data from the database or validating data in the background which is not visible to the website user. Actions like buttons being pressed can change the state of the system.

The class diagram description:

The class diagram represents the system model.

We have userController is the parent of 3 different types of users which are adminController, companyController, and jobSeekerController that they inherit all the attributes and the methods from userController. Each user is different and this is specified by their constant role attribute in the database.

Each admin can add from one to many companies.

Each company can add and manage many job posts.

The sequence diagram:

The sequence diagram shows the interactions between the users of the system (actors) and the system controllers/views/database.

The users can only interact with the views by or pressing buttons but the actual views then interact with the controllers to perform the desired action of the user when the buttons are pressed. If the is data handling, The system controller interacts with the database to retrieve/add/delete data.

Design & Implementation:

- Design Decription:

  The design began with the modeling diagrams such as:

  1. Use-Case diagram.
  2. Sequence Diagram.
  3. Class Diagram.
  4. State Diagram.
  5. Activity Diagram

  Afterwards, the system architecture and application architecture were defined followed by choosing the appropriate language and framework to build the website.

  Afterwards, the requirements table was built as a starting point. The Agile Proccess Model was used therefore the requirements table was created initially and it was updated when it was needed by changing or removing some requirements during development.

  Automated testing was applied on some functionalities like the Register functionality.

- Implementation:

  To build the backend, the Laravel Framework was used to build the website using PHP, Blade and Laravel's Eloquent was used as an object relational mapper (ORM) that is included by default within the Laravel framework. An ORM is software that facilitates handling database records by representing data as objects, working as a layer of abstraction on top of the database engine used to store an application's data.

  For the Frontend, we used HTML and CSS.

  During implementation, the development phases plan was followed and the work was divided equally between all team members.

  Visual Studio Code was used to develop the whole project and GitHub was used for version control.

  When the stage of implementing each user (Job Seeker,Company Admin, Website Admin) was reached, each team member created a separate branch in the repository to implement the needed functionality and then the branches were merged together after all the tests have passed.

REQ1 in the functional requirements was our first priority because the whole application depended on it. Most the testing and phases depended on the database. Therefore, we applied incremental development to the database as we drafted an initial schema and used Laravel to create the models through migration.

There are four tables (Users, Applications, JobPost, JobCategory), we wanted to use a composite priority key in the Applications table. Laravel did not support primary composite keys therefore we changed the table and created a new primary key.

Testing:

Development Testing:

We used Laravel JUnit testing to apply automated testing by constructing a few test cases for the following:

Register functionality.

- A function that tries to register a user through the register function in the users controller, it returns true if the registration was successful and false if not.
- A function that sends a user without a name field to the register function.
- A function that sends a user with password field and confirm password field not matched to the register function.
- A function that sends a user with a wrong email format to the register function.

Database:

- A function that sends an email to the database if present it returns true if not it returns false. This is used inside the register function in the users controller because we check if the user is already registered first before we save his information.

Add Company functionality:

- A function that sends company info to the company form and it returns false if the information is not correct and true if the registration was successful.

Add Post functionality:

- A function that sends job post information to the add post form. It returns success if the operation was successful and false if the information was invalid/incorrect.

Release Testing:

The whole system was tested by running all the test cases again and the database was checked after every action that interacts with it. The buttons were also checked that they performed their expected functionality. We tested the website on multiple web browsers (Chroma, Safari, Moicrosoft Edge,.. etc).