IST 412 Team 5

Team Diagrams

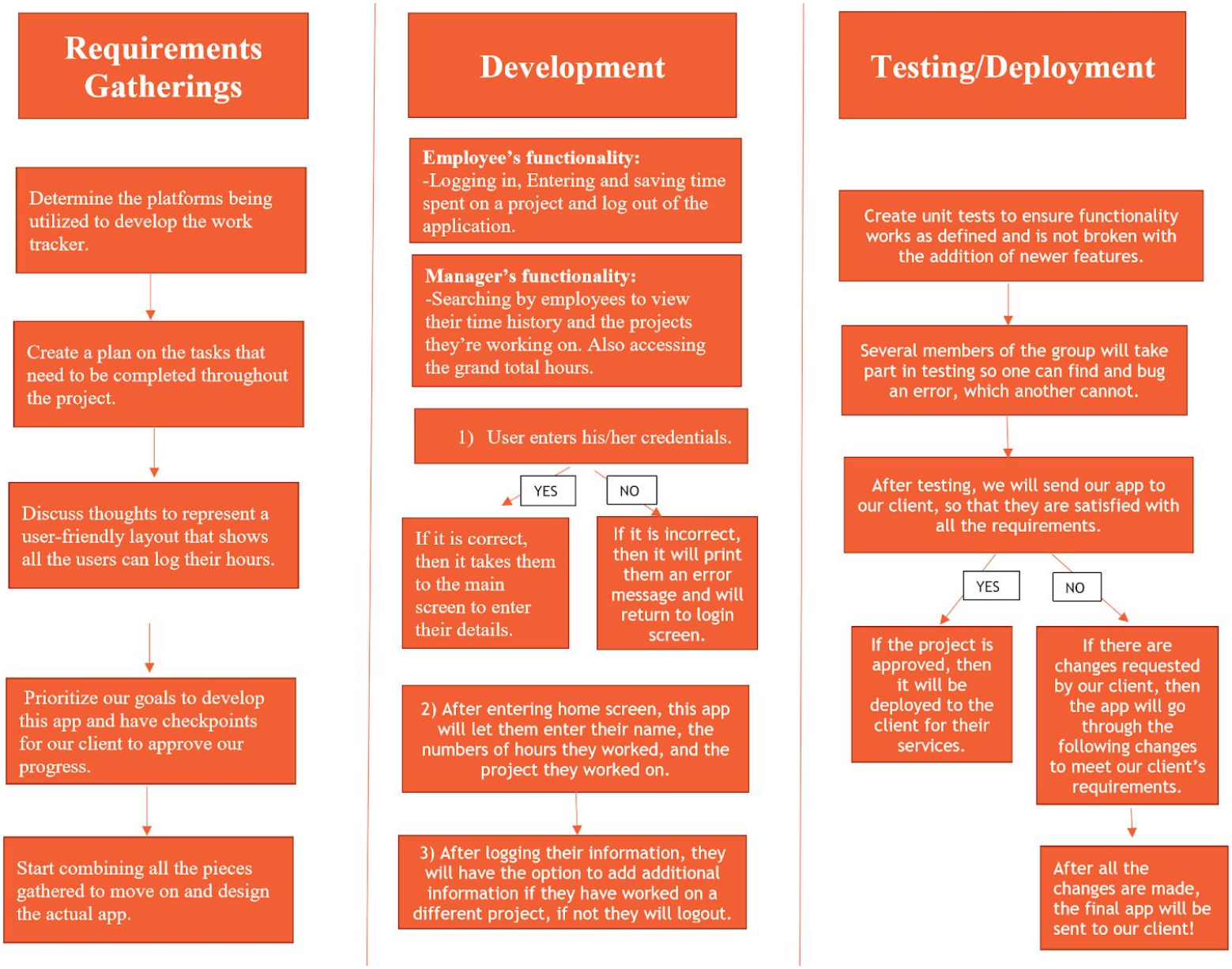
Note: Please see the "Supporting Documentation" section for explanations on the diagrams

Project: Work Hour Tracker

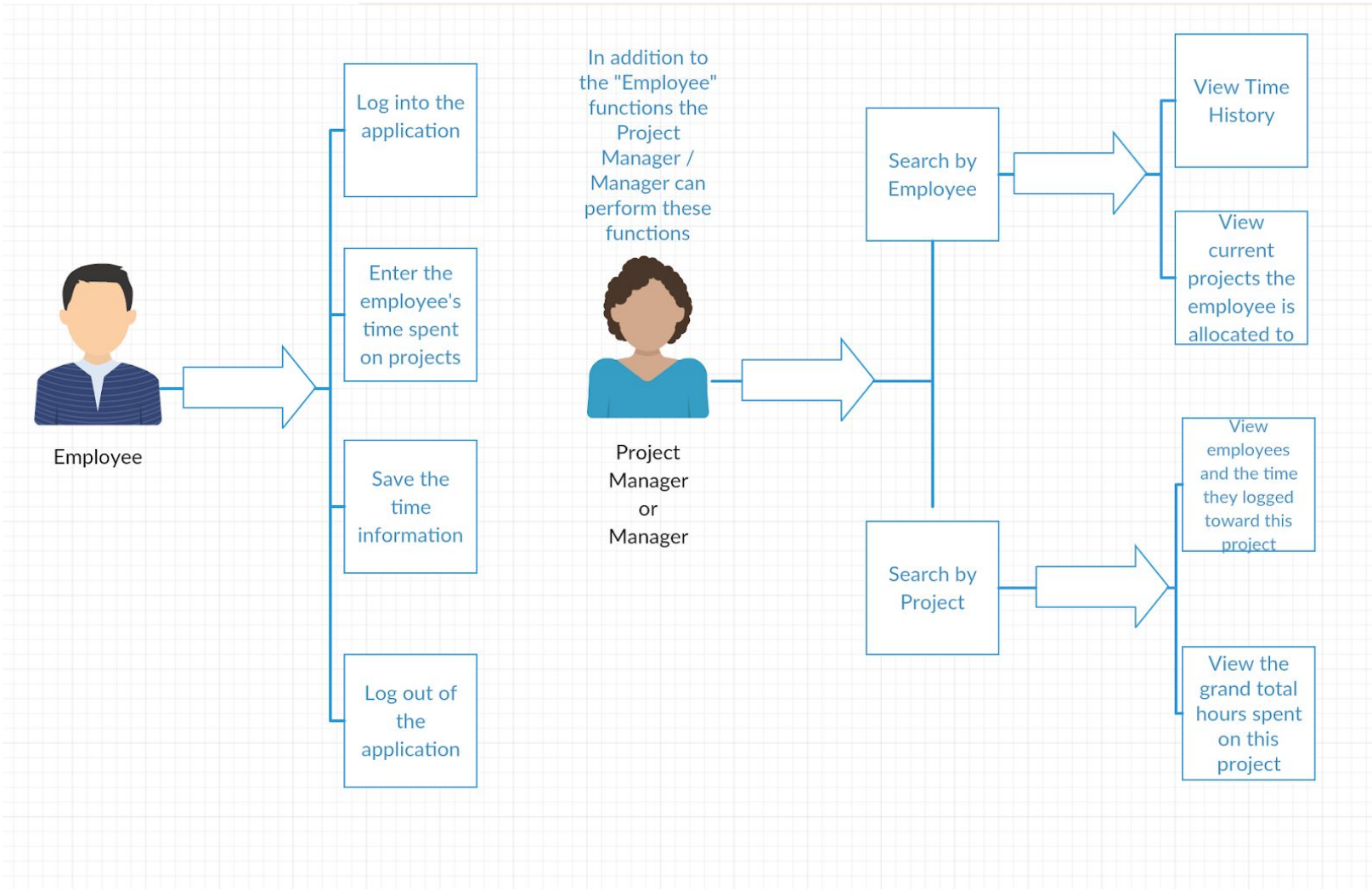
Client: Jim Clarke

Developers: Dennis Smith
Sai Nithisha Guntamadugu
Ricky Zhao
Jackson Penning
Bijal Patel

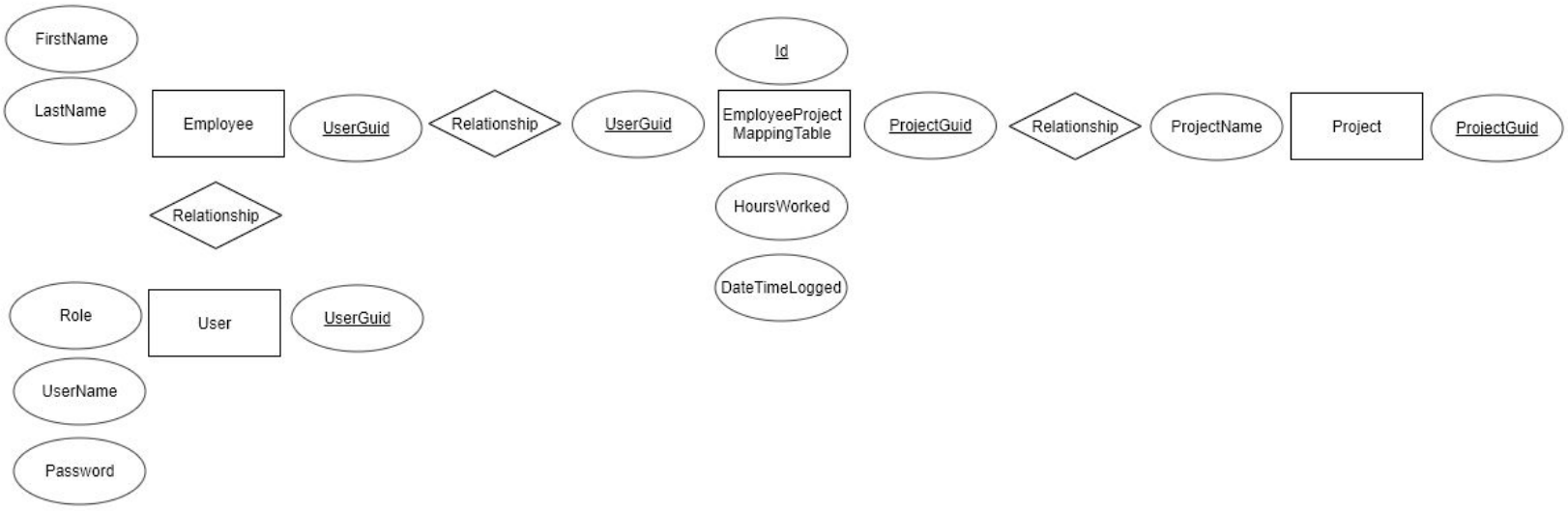
Design Plan


Requirements Gatherings
Determine the platforms being utilized to develop the work tracker.
Create a plan on the tasks that need to be completed throughout the project.
Discuss thoughts to represent a user-friendly layout that shows all the users can log their hours.
Prioritize our goals to develop this app and have checkpoints for our client to approve our progress.
Start combining all the pieces gathered to move on and design the actual app.
Development
Employee's functionality:
-Logging in, Entering and saving time spent on a project and log out of the application.
Manager's functionality:
-Searching by employees to view their time history and the projects they're working on. Also accessing the grand total hours.
1) User enters his/her credentials.
YES
NO
If it is correct, then it takes them to the main screen to enter their details.
If it is incorrect, then it will print them an error message and will return to login screen.
2) After entering home screen, this app will let them enter their name, the numbers of hours they worked, and the project they worked on.
3) After logging their information, they will have the option to add additional information if they have worked on a different project, if not they will logout.
Testing/Deployment
Create unit tests to ensure functionality works as defined and is not broken with the addition of newer features.
Several members of the group will take part in testing so one can find and bug an error, which another cannot.
After testing, we will send our app to our client, so that they are satisfied with all the requirements.
YES
NO
If the project is approved, then it will be deployed to the client for their services.
If there are changes requested by our client, then the app will go through the following changes to meet our client's requirements.
After all the changes are made, the final app will be sent to our client!

User Diagram

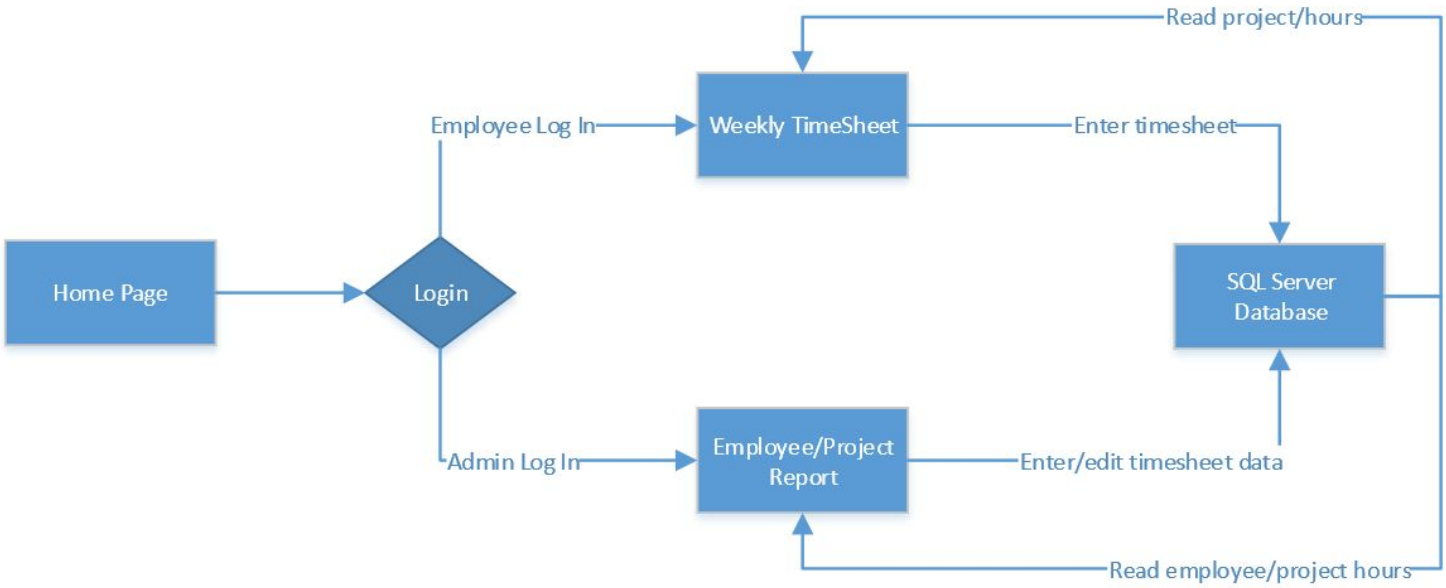


Entity Relationship Diagram

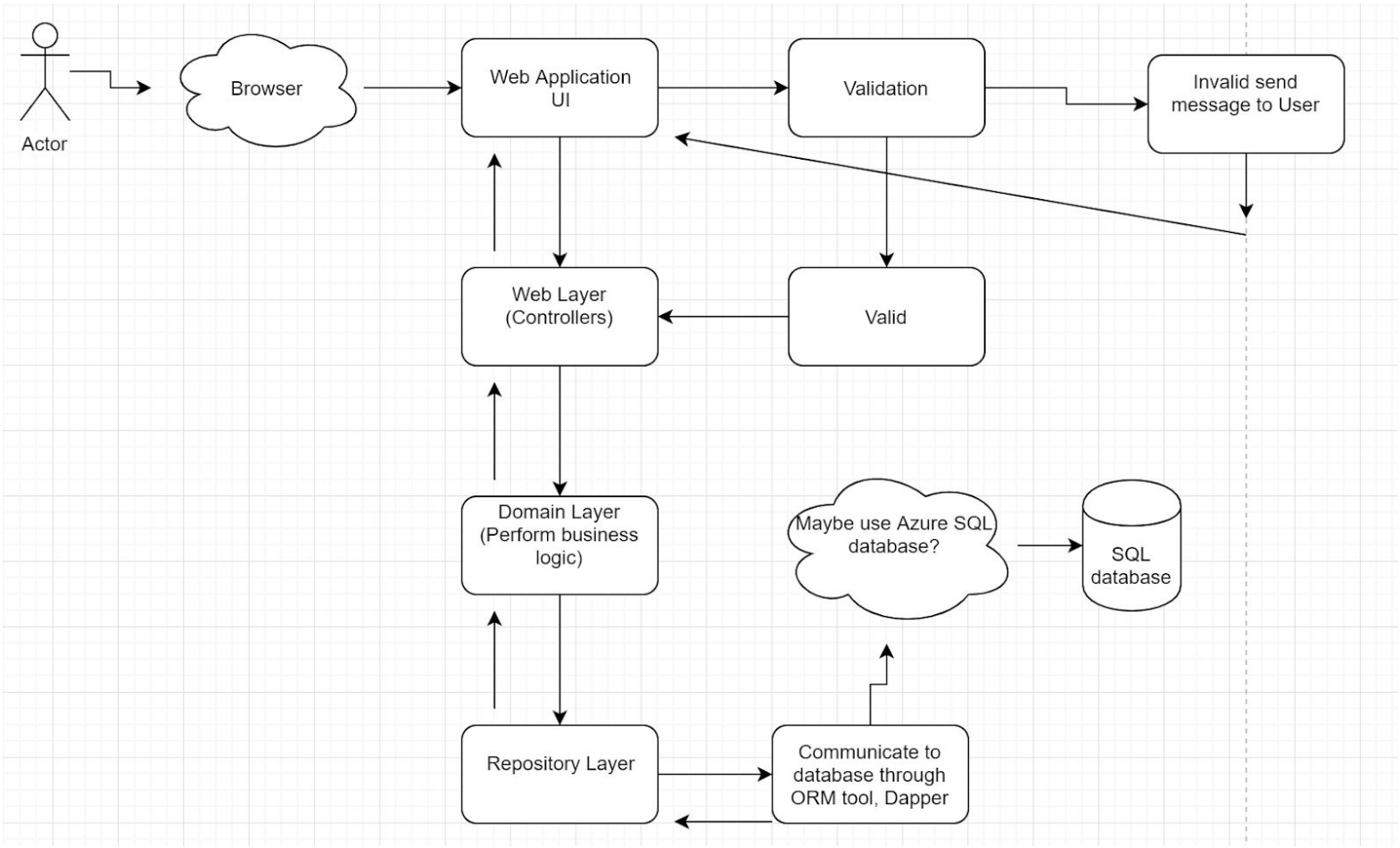


Application Workflow

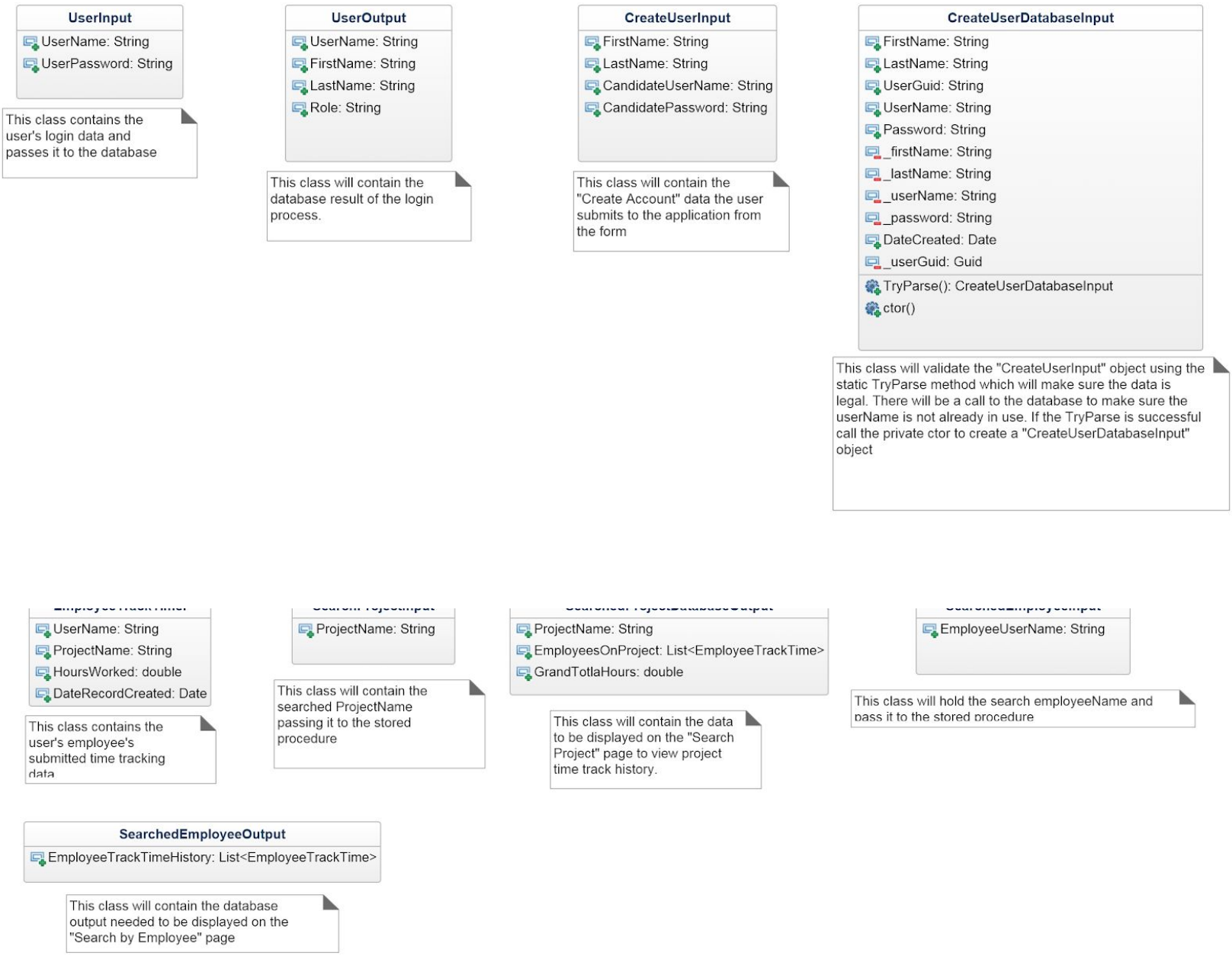
Basic Workflow



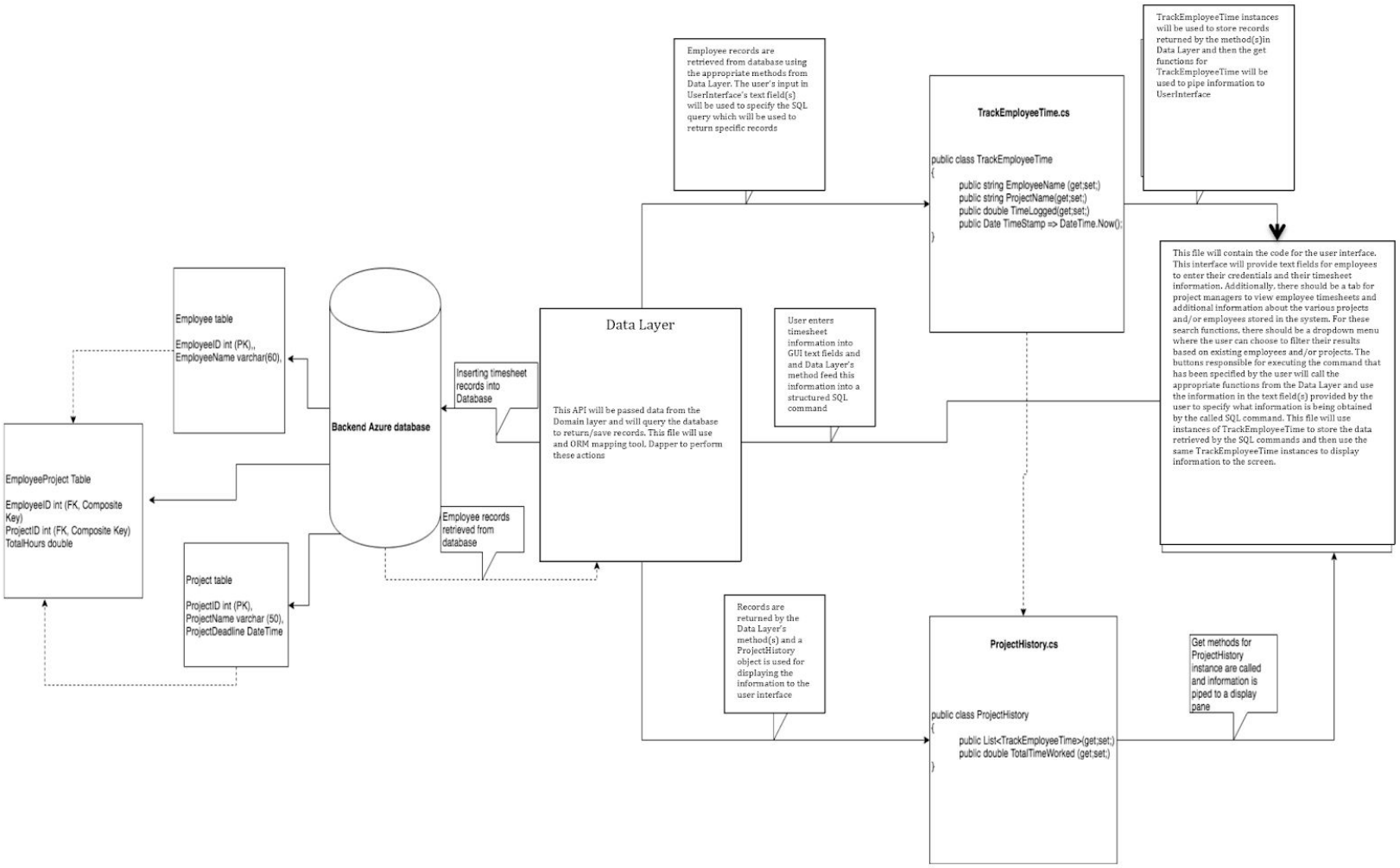
Project level Logic Flow



Class Diagram



Internal Logic Flow Diagram



Supporting Documentation

Design Plan Diagram

This diagram depicts the three phases of the application and what steps are going to be taken during each phase. The Development and Testing phases will occur at the same time as we will be creating unit tests to ensure the application's functionality is not comprised by newer code additions.

User Diagram

The user diagram defines the functionality the lowest level user (Employee) will have access to which includes: logging into the application, entering time on project data, saving that data, and logging out. In addition to those basic functions the "Project Manager / Manager" role will have access to "Employee Search" where they can view the employee's logged time and view a list of projects the employee is working on, "Project Search" where they can view employees who worked on the searched project and the time data they logged toward it as well as view a grand total hours worked on the project across all employees, and "Assign a Project" where they can assign a project to an employee to begin working on (this function is not shown in the user diagram).

Database Design

The database will be a SQL database and will feature 4 tables in total; Employee, User, Project, and EmployeeToProjectMapping tables. There will be a 1-to-1 relationship between the Employee and the User tables. An employee can have many projects and many projects can have many employees. Therefore; there will be a many-to-many relationship between the Employee and Project tables. The EmployeeToProjectMapping table will contain the hours the employee entered for the project, a user entered comment, and the date when the record was created (captured in the application and passed to the database).

Basic Workflow Diagram

This diagram displays the basic workflow from logging in to accessing / saving data.

Project level Logic Flow

This diagram gives an overview of the application's architecture and defines the flow of the work amongst the different layers. The application will be an ASP.NET Core MVC web application created with Visual Studio 2017. The application will feature a four tiered design. The first tier would be the Web layer which will contain the MVC Controllers that will receive user requests from the UI. The next tier will be the Domain layer which will be called from the Controllers and perform business logic on the data. The third tier will be the Data layer which will be passed a request from the Domain layer and will use an ORM (Object Relational Mapper) tool, Dapper, to access the database to fulfill the request and pass it back up through the layers. The fourth tier will be the Model layer which will contain the DTOs needed to pass data to and from the layers. Each of the tiers will be separate projects which would make them APIs that communicate and pass data to one another. The Domain, Data, and Models layers will be .NET Core class libraries and the Web layer will be an ASP.NET Core MVC web project. In addition, there will be unit test projects for the Domain and Web layers (if possible, there will be one for the Data layer too).

Class Diagrams

The application will feature POCO classes that will hold the data needed for UI display. UserInput.cs will be a class that will hold the imputed user credentials which will be verified in the database. UserOutput.cs will contain the data about the user such as: UserName, FirstName, LastName, and Role. The Role property will be used to restrict what pages the user can view. An example using Razor syntax would be if(!@user.Role == "Manager") in order to display or hide a link to a page. Next, I will explain the objects needed for creating an account. CreateUserInput.cs will contain the data the user will submit to the application from the "Create an Account" page. The main goal of that object is to transform it into the "CreateUserDatabaseInput" object. To do this we will call the "CreateUserDatabaseInput" class' static TryParse method which will output a valid CreateUserDatabaseInput object. Next, the EmployeeTrackTime class will be used to hold the track time data submitted from the the form to the application and will be persisted in the database. The SearchProjectInput class will be used to hold the project name the Project Manager / Manager requested through the UI. The string value will be passed to a stored procedure in the database and if there are records it will return them and the ORM mapper will map them to a "SearchedProjectDatabaseOutput" object which will contain the project's name, grand total time spent, and a list of "EmployeeTrackTime" objects. Lastly, the "SearchEmployeeInput" class will pass the user's requested employee userName to a stored procedure and the ORM mapper will map the result to a "SearchedEmployeeOutput" object which will contain a List of "EmployeeTrackTime" objects.

Internal Logic Flow Diagram

The Internal Logic Flow diagram describes the methods the projects in the application will use to communicate with each other at a high level. The data entered by the user from the UI will be passed from the MVC controller to the Domain layer which will pass it to the Data layer where Dapper (the ORM tool we are using) will perform the necessary database query to fulfill the user's request and depending on the request (such as retrieving data for display) it will pass the data back up through the layers.