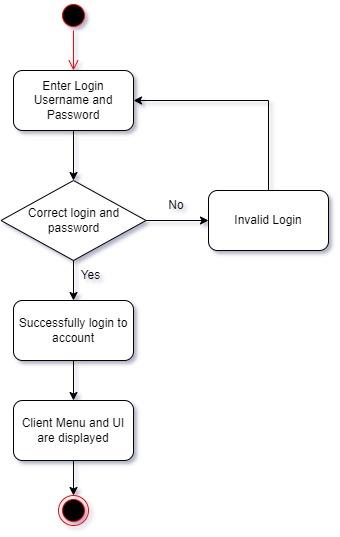
## **4.1** **Class Diagram**

*A detailed class diagram is developed to depict the relationships between classes as well as the attributes and operations of each class. Provide detailed description as well.*

## **4.2** **Activity Diagram**

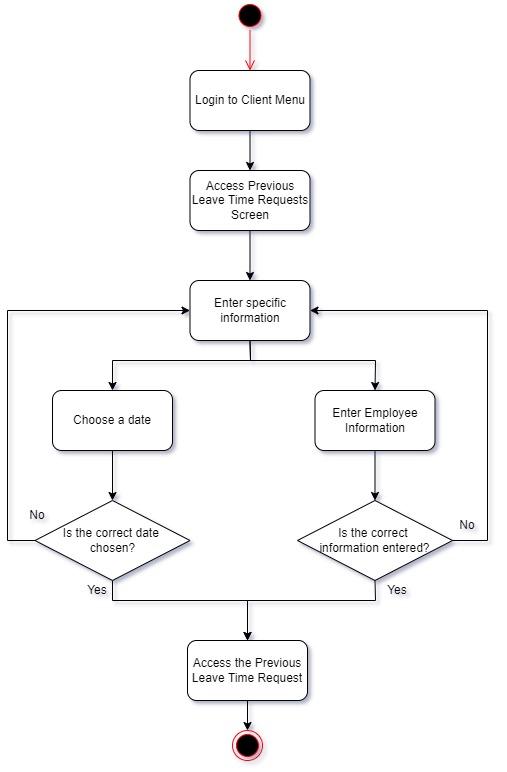
*Develop activity diagrams for the non-trivial methods. Provide detailed description for each diagram as well.*

The Vacation Tracking System (VTS) design involves seven detailed activity diagrams for the essential use cases that address several key features and actions that can occur in the system. The first activity diagram handles the workflow for logging into the system. The activity diagram demonstrates a use case that applies to all types of users as everyone accessing the system must log in to the existing intranet portal system using their credentials. When the user enters to their login username and password, the system will check if the information entered is correct and in the system. If the information was correct then the screen will demonstrate that the user logged in successfully and the client user interface will be displayed. If the information entered is incorrect, then the screen will display the information that was not correct and invalid. The system should then bring the user back to the login screen so they may be able to enter their information again.



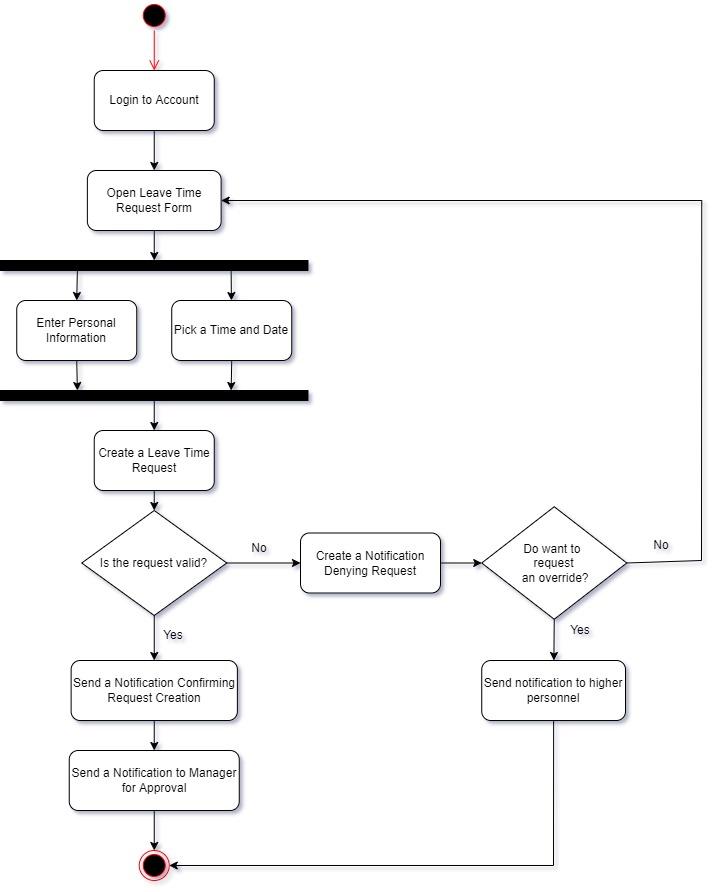
**Figure ?: Activity Diagram for the Login Use Case**

The second activity diagram handles the workflow for access to previous requests made in the previous calendar year. The activity diagram demonstrates a use case that applies to multiple types of users as users with a certain level of approval can access previous requests. After the user logins into the system, the user can access previous leave time requests. Upon clicking the button the user may choose a date or enter an employee’s information (like their name or ID number). The system will then check if the information is correct. If the correct information was entered or chosen then the user will access the specific request. If the correct information was not entered or chosen, the user will be taken back to the previous screen. The user can then try to enter the correct information again.



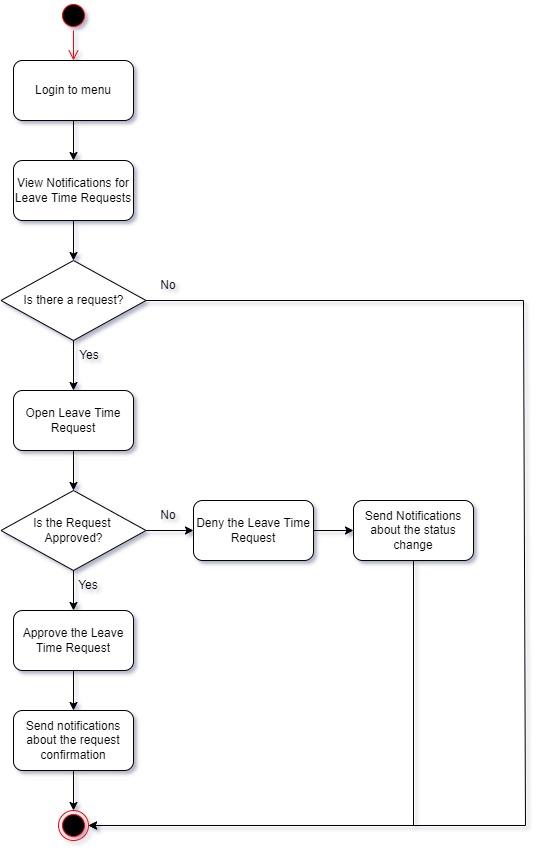
**Figure ?. Activity Diagram for Accessing Previous Leave Time Requests**

The third activity diagram handles the workflow for creating a leave time request. The activity diagram demonstrates the most common use case that primarily applies to employees as they usually just send the request as the first option to get time off for any reason. After the user logs into the system, the user can then open a leave time request form. The form will prompt the user to enter their information and pick the time and date for the amount of leave time. All of the entered information must be entered to proceed. Afterward, the system will then try to create the leave time request but the system checks if the request is valid. Specifically, the system will check if the request is valid and verified. If the request is valid then the system will send a notification to the user that the request was successfully created and submitted. The system will then send a notification to the manager to state that a request needs approval. If the request is not valid then the system will create and send a notification of the request denial. The system will prompt the user and ask if they need an override to create their request. If the user does want an override, then the system will send a notification to higher personnel like HR and the system administrator asking for an override. If not, the user will be taken back to the screen asking for the information.



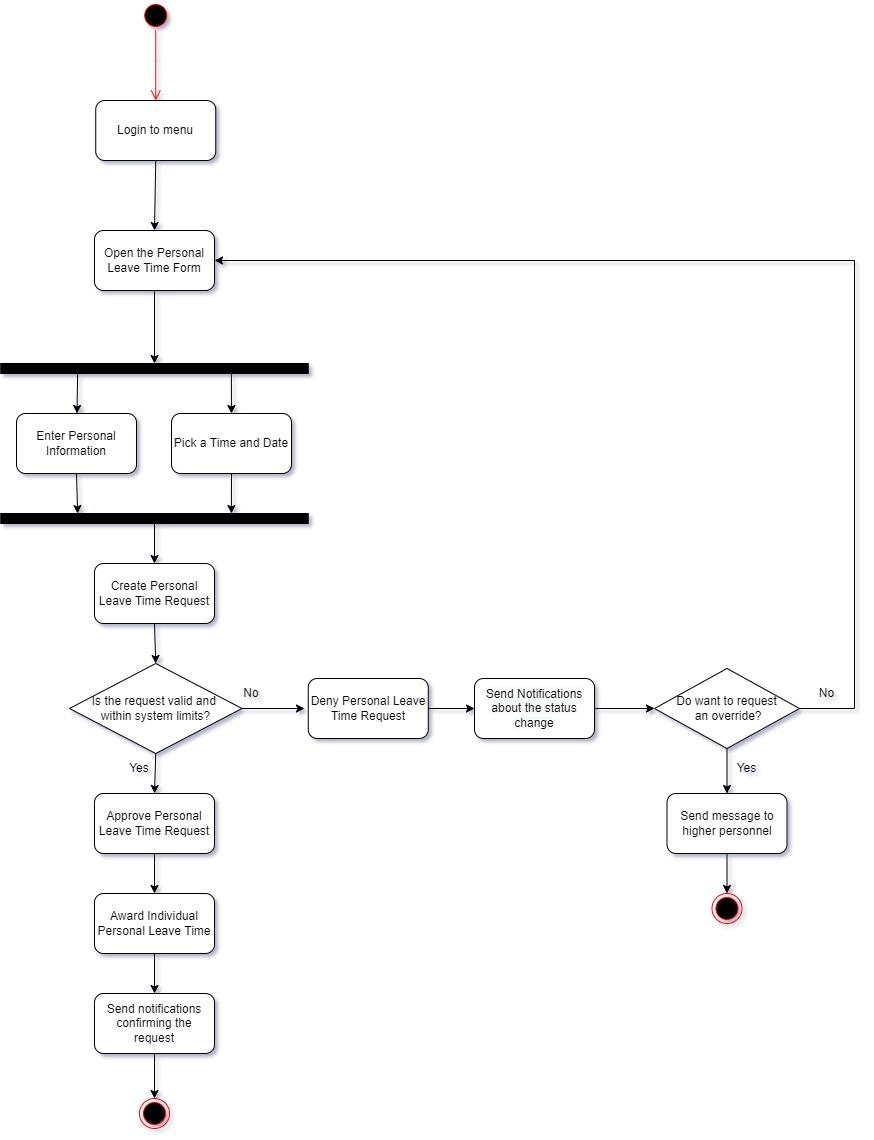
**Figure ?. Activity Diagram for Creating a Leave Time Request**

The fourth activity diagram handles the workflow for manager approval. The activity diagram demonstrates a common use case that applies to managers as they are usually the people that approve the most requests. After the user logs into the system, the manager may receive and view notifications for leave time requests that need approval. If there is not a request then the manager can continue with what they were doing. If there is a request, then the user can open the leave time request. The user can then read and evaluate the request and choose to approve it. If the user chooses to approve the request, then they can approve it and the system sends notifications about the request confirmation. If the user does not approve the request, then the user can deny the leave time request and the system then sends notifications about the status change.



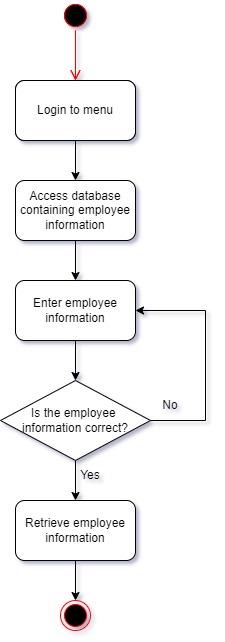
**Figure ?. Activity Diagram for Manager Approval**

The fifth activity diagram handles the workflow for awarding personal leave time. The activity diagram demonstrates a use case that applies to managers as they have the ability to directly award personal leave time (within system-set limits). Once the user logs in to the system, they can open a personal leave time form. The user then needs to enter the personal information of the employee who is being awarded the personal leave time and pick a time a date for how long the leave time is. All of the information must be entered before continuing the process. Once the user enters the necessary information, they may try to create the personal leave time request. The system will then check if the request is valid and within system-set limits. If the request is valid, then the system approves the personal leave time request and then awards the individual their personal leave time (which should be seen in their user interface). Finally, the system will send notifications confirming the request. However, if the request is not valid then the system will deny the personal leave time request and send notifications about the request’s status change. The system will prompt the user and ask if they request an override for the leave time request. If the user does request an override, then the system will send a message to higher personnel like HR or a system administrator. If the user does not want to request an override. Then the system will take the user back to the screen that prompts the user to enter the necessary information again.



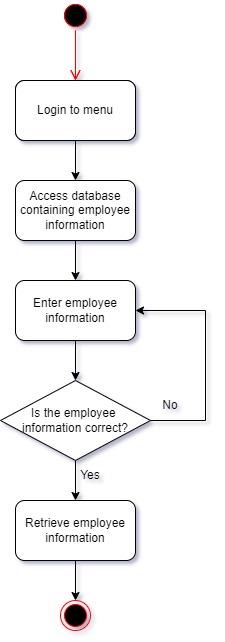
**Figure ?. Activity Diagram for Awarding Personal Leave Time**

The sixth activity diagram handles the workflow for overriding actions that are restricted by rules. The activity diagram demonstrates a use case that applies to personnel like HR and system administrators as they have the ability to override actions restricted by the rules-based system and allow the leave time request to be awarded. Once the user logs in to the system, the user may receive and view notifications for override requests that need approval. If there is not a request then the administrators can continue with what they were doing. If there is a request, then the user can open the override request. The user can then read and evaluate the request and choose to approve it. If the user chooses to approve the request, then they can approve it and the system sends notifications about the request confirmation. If the user does not approve the request, then the user can deny the leave time request and the system then sends notifications about the status change.



**Figure ?. Activity Diagram for Overriding Actions**

Finally, the seventh diagram handles the workflow for accessing employee information and changes. The activity diagram demonstrates a use case that applies to HR as their interface is integrated with the HR department legacy systems and they have the ability to retrieve the required employee information and changes. Once the user logs in to the system, they can then access the database containing employee information. The user can then enter the employee’s information like their employee ID or name. The system will then check if the employee information is in the system or if the correct employee was picked. If the information is correct, then the user can retrieve the employee information and the changes that have occurred. However, if the information is incorrect then the user can go back to the screen and try to enter the employee information again.



**Figure ?. Activity Diagram for Accessing Employee Information and Changes**

## **4.3 UML Message Sequence Diagram**

## **4.4 UML Statechart Diagram**

## **4.5** **Algorithms for Components/Methods**

*Detailed pseudo-code for the non-trivial methods should also be developed at this stage. {Set, Get, default constructors, etc., are considered trivial methods where pseudo code is not required.}*

## **4.6** **Database Design**

Provide a description of any tables, fields, and relationships that need to be created for the design. ERDs also should be developed. Primary keys, foreign keys, and alternative keys, integrity rules and constraints should also be stated if applicable.

The database design for the VTS system is extensive as it has 10 tables containing information regarding the different users (employee, manager, HR, and system admin), the various forms regarding leave time, and notifications. The *Employee* table is meant to represent an employee and contains several attributes that are meant to behold information for each employee. Specifically, the table contains attributes about the employee’s ID, name, job title, possible changes, email, and login information. There is only one primary key for the table as it acts as a unique identifier. However, there is a primary key constraint (PK\_Employee) that states the primary key value is made up of the ID and email attributes.

**Figure?. Employee Table and Relationship(s)**

The *Manager* table is meant to represent a manager and contains a few attributes specifically for actions that a manager can do and may need a record of. The table contains attributes about the manager’s ID, references to the employee ID and request ID, and the approval value. The manager ID is the primary key for the table as it acts as a unique identifier. The employee ID is the foreign key for the table and refers to the primary key of the *Employee* table. The *Manager* table and the *Employee* table have a one-to-many (1:M) relationship as one manager has many employees reporting to them. The request ID is another foreign key for the table and refers to the primary key of the *Leave Time Request Form* table. The *Manager* table and the *Leave Time Request Form* table have a one-to-many (1:M) relationship as one manager is handling multiple leave time request forms.

**Figure?. Manager Table and Relationship(s)**

The *HR* table is meant to represent an HR representative and contains a few attributes specifically addressing the actions that an HR representative can do and may need a record of. The table contains attributes about a representative’s ID number, references to the employee ID and request ID, override status on a request, and a report containing employee information and changes. The HR ID is the primary key for the table and acts as the table’s unique identifier. The employee ID is the foreign key for the table and refers to the primary key of the *Employee* table. The *HR* table and the *Employee* table have a one-to-many (1:M) relationship as one HR representative is responsible for multiple employees and one employee. The request ID is another foreign key for the table and refers to the primary key of the *Leave Time Request Form* table. The *HR* table and the *Leave Time Request Form* table have a one-to-many (1:M) relationship as one HR representative is responsible for handling multiple leave time request forms.

**Figure?. HR Table and Relationship(s)**

The *System Administration* table is meant to represent a system administrator and contains a few attributes specifically addressing the actions that an HR representative can do and may need a record of. The table contains the administrator’s ID number, the override status on a request, and references to the employee ID and request ID. The administration ID is the primary key for the table and acts as the table’s unique identifier. The employee ID is the foreign key for the table and refers to the primary key of the *Employee* table. The *System Administration* table and the *Employee* table have a one-to-many (1:M) relationship as one system administrator is responsible for handling multiple employees (like a system issue). The request ID is the foreign key for the table and refers to the primary key of the *Leave Time Request Form* table. The *System Administration* table and the *Leave Time Request Form* table have a one-to-many (1:M) relationship as one system administrator will typically handle multiple leave time request forms.

**Figure?. System Administration Table and Relationship(s)**

The *Leave Time request form* table is meant to represent the form an employee needs to submit in order to request leave time. The table contains several attributes that represent the information involved in submitting the form. The table contains a request ID number, the request start/end date and time to state how long the leave time is, the type of leave time (i.e. personal time off or sick leave), a small description for the leave time reason, if the request required an override, a summary for the request, and a reference to the employee ID. The request ID is the primary key for the table and is the unique identifier. The employee ID is the foreign key for the table and refers to the primary key of the *Employee* table. The *Leave Time Request* form table and *Employee* table have a one-to-many (1:M) relationship as one employee can submit multiple leave time request forms.

**Figure?. Leave Time Request Form Table and Relationship(s)**

The *Activity Log* table is meant to represent the activity logs that are for all of the transactions that can occur. The table contains a few attributes that represent the information involved in each transaction. The table contains a transaction ID, the date of the transaction, the time of the transaction, the transaction summary, and a reference to the employee ID. The transaction ID is the primary key and unique identifier for the table. The employee ID is the foreign key for the table and refers to the primary key of the *Employee* table. The *Activity Log* and *Employee* tables have a one-to-many (1:M) relationship as an employee can have multiple activity logs as they complete several transactions.

**Figure?. Activity Log Table and Relationship(s)**

The *Email Notifications* table is meant to represent the email notifications that result from the various requests that can occur. The table contains a few attributes that represent the information involved in each notification. The table contains a notification ID, the date of when the notification was sent, the when the time of the notification was sent, the notification message, and a reference to the employee ID. The notification ID is the primary key and unique identifier for the table. The employee ID is the foreign key for the table and refers to the *Employee* table primary key. The *Email Notifications* and *Employee* tables have a one-to-many (1:M) relationship as an employee can have and/or send multiple email notifications.

**Figure?. Email Notifications Table and Relationship(s)**

The *Personal Leave Time Form* table is meant to represent the form that directly awards personal leave time to the employee. The table contains the personal leave time ID, the start/end date and time, the type of leave, a summary of the personal leave time, if the request asked for an override, and a reference to the employee ID. The personal leave ID is the primary key and unique identifier for the table. The manager ID is a foreign key for the table and refers to the *Manger* table primary key. The *Personal Leave Time Form* and *Manager* tables have a one-to-many (1:M) relationship as one manager can award multiple employees with personal leave time. The employee ID is a foreign key for the table and refers to the *Employee* table primary key. The *Personal Leave Time Form* and *Employee* tables have a one-to-many (1:M) relationship as an employee can be awarded personal leave time multiple times.

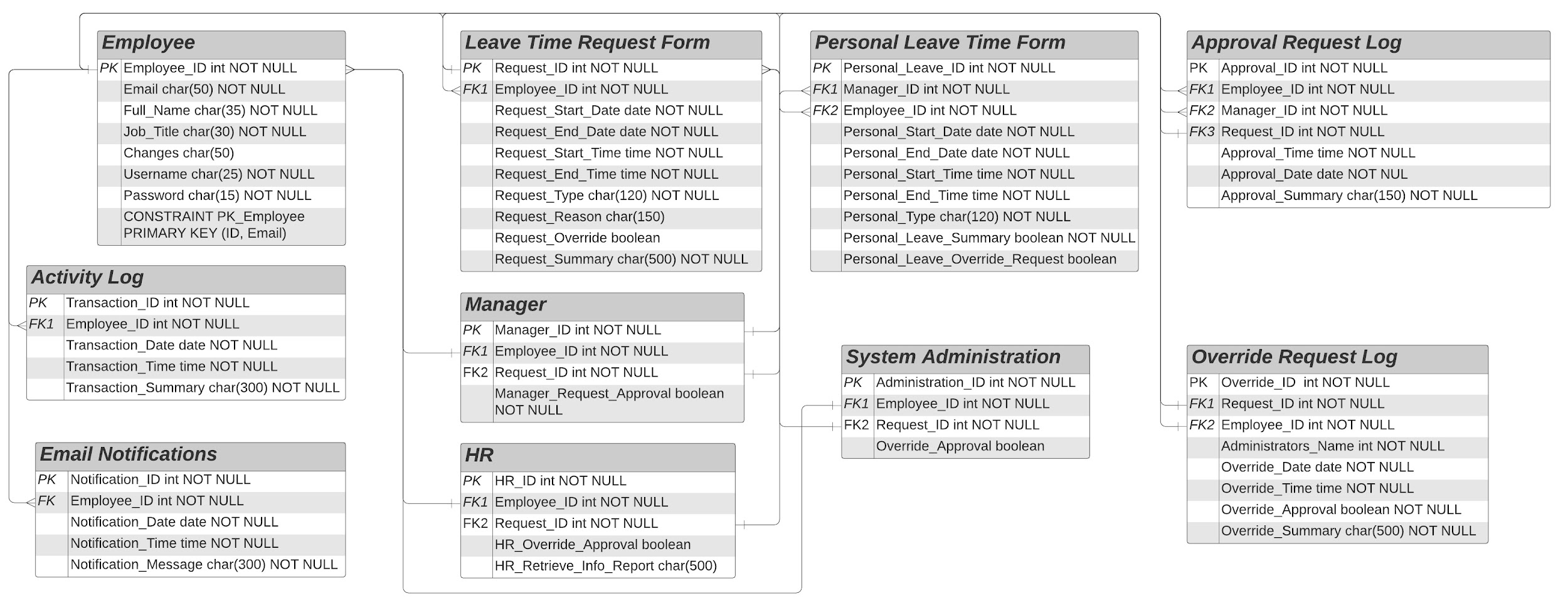
**Figure?. Personal Leave Time Form Table and Relationship(s)**

The *Approval Request Log* table is meant to represent the logs created from an approval request by the manager. The table contains the approval ID, the approval time, the approval date, a summary of the approval request, and references to the employee, manager, and leave time request form tables. The approval ID is the primary key and unique identifier for the table. The employee ID is a foreign key for the table and refers to the primary key of the *Employee* table. The *Approval Request Log* and *Employee* tables have a one-to-one (1:M) relationship as an employee can receive one approval for each leave time request. The manager ID is a foreign key for the table and refers to the *Manager* table primary key. The *Approval Request Log* and *Manager* tables have a one-to-one (1:1) relationship as a manager can approve one leave time request at a time. The *Approval Request Log* and *Leave Time Request Form* tables have a one-to-one (1:1) relationship as one leave time request only need one approval request.

**Figure?. Approval Request Log Table and Relationship(s)**

The *Override Request Log* table is meant to represent the logs created from an override request by administration personnel like HR and system admins. The table contains the override ID, the administrator looking over the request’s name, the date and time of when the override request was evaluated, whether the override request was approved, a summary of the override request, and references to the leave time request form and employee tables. The override ID is the primary key for the table. The request ID is a foreign key for the table and refers to the primary key of the *Leave Time Request Form* table. The *Override Request Log* and *Leave Time Request Form* tables have a one-to-one (1:1) relationship as one leave time request can have the rules be overridden once. The employee ID is a foreign key for the table and refers to the primary key of the *Employee* table. The *Override Request Log* and *Employee* tables have a one-to-many (1:M) relationship as an employee can try to ask for and/or receive multiple overrides for their leave time requests.

**Figure?. Override Request Log Table and Relationship(s)**

****

**Figure?. Overall Database Design**