CSCI 5448 Project Part 4
Final Report

Personal Activity Assistant

Bharat Nallan Chakravarthy
Megha Sree Yadla
Swati Patil

1. What features were implemented?

These were the features that were implemented.

1. Users can Sign Up to the Personal Activity Assistant.
2. Users can Login to their Personal Activity Assistant.
3. Users can add these:
    a. Address - Add, edit and delete Address.
    b. Notes- Add, edit and delete Notes.
    c. Appointments- Add, edit and delete Appointments and restore or reset the system to the default state.
    d. Expenses- Add, edit, delete, export and undo the Expenses.
    e. Reminders- Add, edit and delete Reminders.

2. Which features were not implemented from Part 2?

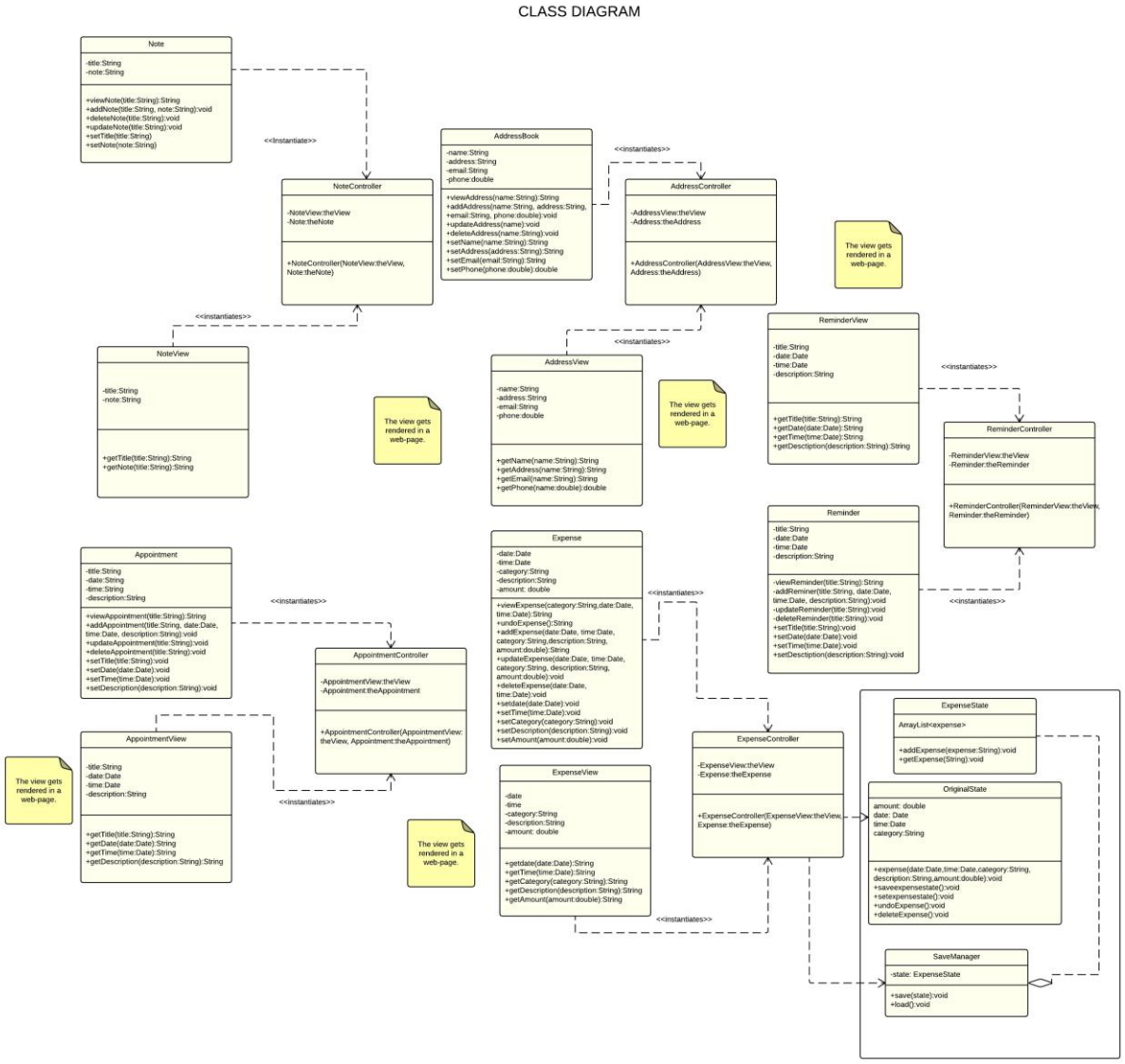
Almost all the features and requirements from Project Part 2 were implemented. One of the features from Project Part 2 that was not implemented was Non-Functional Requirement 02. The requirement was to send a mail to the user's mail ID when the user logs in from a new device. We implemented eight design patterns with MVC architectural pattern so we ran out of time to implement NFR-02.

3. **Show your Part 2 class diagram and your final class diagram. What changed?**
Why? If it did not change much, then discuss how doing the design up front helped in the development.

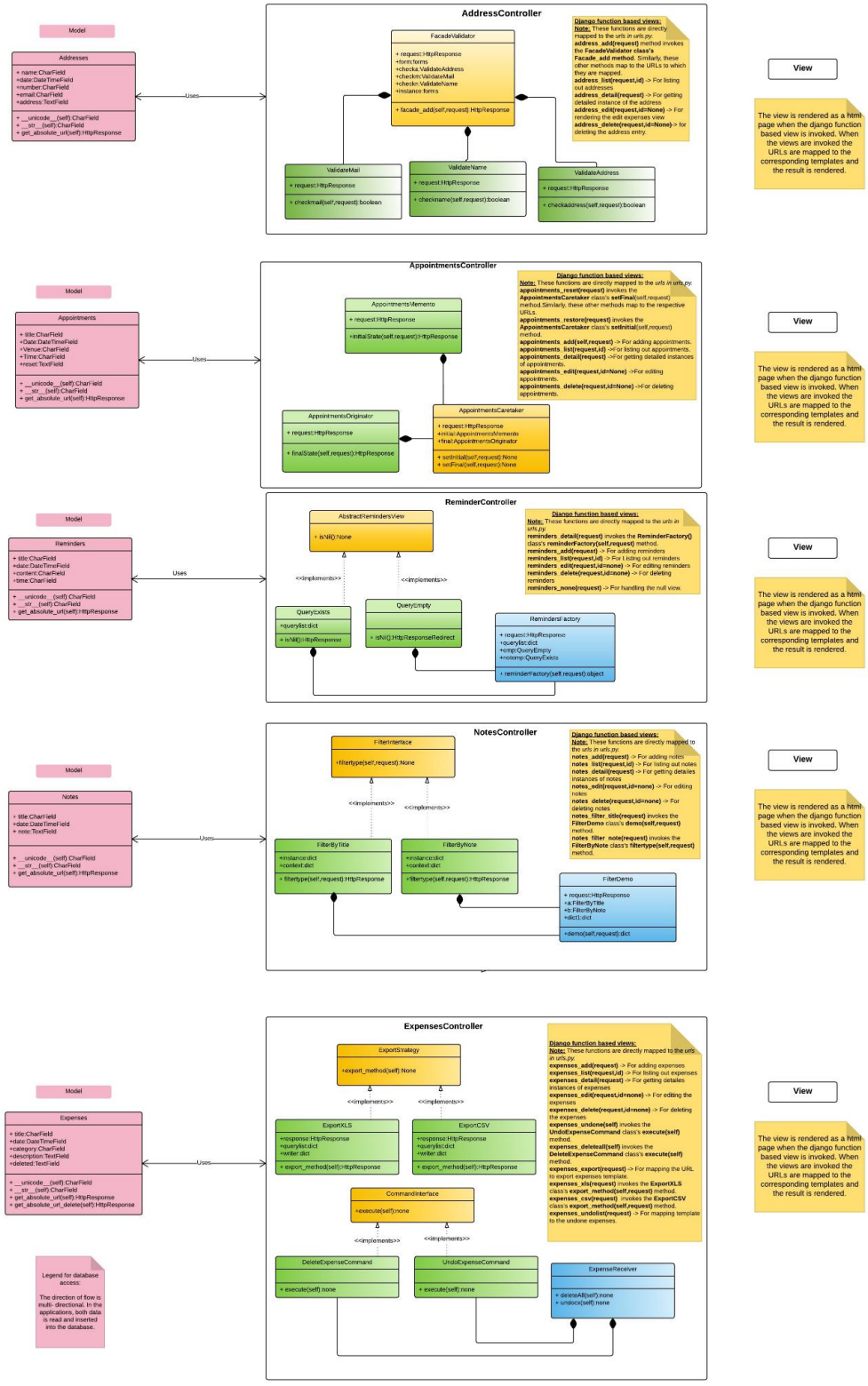
There were a lot of changes in the class diagram. Initially, we had only planned to use 2 design patterns along with the MVC architectural pattern. In the end, we used eight design patterns along with the MVC Architectural pattern.

So, there were more interfaces and abstract classes that were used. This also meant that the entire structure of the class diagrams that we proposed changed. There, were a lot of interfaces and inheritances that were implemented in these 8 patterns. This made our class diagrams much more complex.

The class diagram for part 2 is as shown below.



The class diagram for part 3 is as shown below.



4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

We had initially planned to make use of the Command and the Memento Design Patterns along with the MVC architectural pattern.

In our Final Project, we were able to use up to 9 design patterns including those proposed.
The patterns that were used and their application context is as shown below.

DP. No	Name of Pattern/ Used in Application	Purpose
1.	Model View Controller/ Overall Application	For organizing the project according to Model, Views and Controllers
2.	Command/Expenses Application	Used Command Design pattern to undo a deleted expense and also to permanently delete it from the trash.
3.	Memento/Appointment Application	Used Memento to restore the state of the appointments application based on initial or final states.
4.	Strategy/Expenses Application	Used Strategy to export the expenses in two different formats. Each Strategy abstract base class consisted of its own algorithm - which was used to export the expense in XLS or CSV format.
5.	Facade/AddressBook Application	Facade was used to check all the Data Fields to see if they already exist in the database. For instance, this was used to check if the current address already existed in the Address Book.
6.	Iterator/Notes Application	Used the iterator design pattern along with the Django iterator plugin to create pagination. This was used to limit the number of entries that were displayed per page and allowed to loop through the pages.
7.	Null-Object/Reminders Application	In any instance, when the given query is empty, the screen displays no entries. This may be confusing a lot of times. To handle this, the Null-Object pattern was used. This actually handles the view accordingly.
8.	Filter/Notes Application	The filter design pattern was used to search for notes using either the note content or the note title.
9.	Proxy/Overall Application	The overall application uses the Proxy design pattern. This happened deliberately. We used Object relational mapping to

		map the form entries to the databases - since this was not done directly, but rather through external classes, this formed the Proxy design pattern in our application.

5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

The course throughout gave us a clear understanding about how to go about a software development process and how effective it is to design before coding using various design patterns.

1) Through the process of Analysis and Design, we learned to implement and design highly maintainable systems. These systems are not only well organized, they are also easy to modify.
2) We learnt that following the process of first designing and then implementing the software helps us to incorporate any requirement changes at a later phase.
3) Discussing the design within the team gave us some amazing ideas and features to implement in the project.
4) We learnt the different UML diagrams, notations and their importance in the process of analyzing and designing software.
5) Throughout the course we learnt the importance of designing before implementing as it helped us avoid any mistakes in the implementation.
6) It taught us how following a design pattern helps us solve the problems faced in software development.
7) We can reuse the design in some other project where we are supposed to implement similar features.
8) Also we can reuse the design pattern with some modifications, if needed, to solve a similar problem as patterns are best-practice solutions for recurring problems in a particular context.