Design Document #3

A list of design decisions and explanations about how our code has improved since Phase 1.

Preface: Several refactors and features were implemented in this phase but there were no wide-scale design decisions that affected all aspects of the code in phase 2. This is because our design in phase 1 was extensible and did not require expansive refactors.

- **[Feature] Clinic Availability:**
  - Description
    - Clinic availability is a feature where clinics can set and update the times throughout each week that they are available for patients to visit.
    - Users can freely view a clinic's updated availability through the program interface.
  - Decision and improvements
    - Allowing patients to view a clinic's hours from within the app improves the information accessibility between a clinic and its users.
    - Clinics being able to update their hours improves their communication with their user base.
  - Functionality:
    - An admin can set the time block in which the clinic is operating or remove the availability of the clinic on a given day.
    - Select controllers can view clinic availability by calling 'view clinic info'.
- **[Feature] Doctor/Patient Appointments:**
  - Description
    - Appointments are a feature where secretaries can book time slots for a patient to visit a specific doctor. These appointments can be viewed by the patient and doctor associated with them.
  - Decision and improvements
    - Allowing appointments to be stored on the program allows users to keep track of all appointments in an organized and readable manner.
    - Automating the process of booking appointments through the program makes the clinic's operations more efficient.
    - Allowing users and doctors to view their appointments and all their details improves the information accessibility between the clinic and its users and employees.
  - Functionality
    - We allow secretaries to book/cancel appointments on a given day between a doctor and a patient given that it doesn't conflict with another appointment and is within the clinic's availability on that day.
    - Patients and doctors can then view their appointments when signed in.

        - Secretaries can view the appointments of any patient or doctor.
        - When booking an appointment, the selected doctor's appointments and the clinic's availability are displayed to the secretary.
- **[Feature] Doctor reports for patients.**
    - Description
        - This is a feature where doctors can write a report on the system for a specific patient user that contains information about patient visits, medical history, medications, etc.
    - Decision and improvements
        - Allowing the doctor to write reports on the program allows doctors and patients to keep track of what was discussed in appointments in a centralized platform, instead of relying on pieces of paper and memory.
        - If a patient switches to another doctor, reports are useful because the new doctor can view the reports that the previous doctor created on the patient's file and have an understanding of the patient's medical history.
    - Functionality
        - Reports are linked to each patient.
        - Doctors can now add reports after they load a patient in their controller, with a header and body.
        - It automatically notes the date and time of the creation and the name of the doctor who created it.
        - Doctors can also remove reports.
- **[Feature] Regex on Account Creation:**
    - Description
        - A feature that uses regular expressions to ensure that users follow a specific convention when registering an account.
        - In order to create a user account:
            - The username must contain 6 characters and be completely alphanumeric (no special characters or spaces).
            - The password must contain 8 characters which can be anything.
    - Decision and Improvements
        - By providing a format that users must follow when creating their account, we improve the security and privacy of user accounts.
    - Functionality
        - An error message is displayed if a user attempts to create an account that does not follow the correct format.
- **[Feature] Levenshtein distance to suggest spellings of invalid user inputs:**
    - Description
        - To increase the accessibility and ease of use of our program, we implemented a feature that corrects the spelling of users if they are off by a few characters.
    - Decision and Improvements

        - This feature satisfies the Tolerance for Error Principle of universal design, allowing our program to be accessed by a wider variety of people.
        - This also makes the program more fluid, it was often frustrating having to retype a command just because one letter was misspelled.
    - Functionality
        - When the user inputs an incorrect command that has similar spelling to an actual command, we suggest a valid spelling of that command and ask them whether it was their intended input. If they respond with 'y', the command will be executed, otherwise their command is not processed.
- **[Refactor] Abstracted user management commands into a separate controller**
    - Description
        - The list of commands relating to an admin's management of users was very long, so we decided to group all commands relating to admin user management (e.g. create doctor) in the User Management Controller.
    - Decision and Improvements
        - Allowing the admin to create/delete users and change user passwords on a separate screen.
        - This decision meant cleaner looking code in the admin controller and the adminUserManagement controller. It also meant that the help menu for an admin user would be much more concise.
    - Functionality
        - Admin can access this new state by typing 'manage users' or by typing the associated number with it in the help command.
        - Admin can back out of this screen by typing 'back'.
- **[Refactor] Abstracted common menu commands into an abstract class**
    - Description
        - The list of commands relating to a menu screen that a user interacts with (e.g. contact management menu) were abstracted into an abstract menu controller class.
    - Decision and Improvements
        - We had repeated code in all of the different menu related classes. For example, we had to append the "back" command to the command list for every menu every time we made a new menu controller.
        - This decision meant removing the duplicate code smell and making our program more extensible in terms of creating new menu screens efficiently.
    - Functionality
        - The MenuController is a subclass of the TerminalController.
        - All controllers relating to menu screens inherit all the commands from the MenuController.
- **[Refactor] Abstracted common commands related to loading in patients into an abstract class.**

    - Description
        - The list of commands relating to actions that can be performed when a user loads a patient (e.g. a doctor loading a patient and giving that patient a prescription) were abstracted into an abstract menu loaded patient controller class.
    - Decision and Improvements
        - We had repeated code in the different loaded patient classes. For example, we had to append the “unload patient” command to the command list for every loaded patient controller.
        - This decision meant removing a duplicate code smell and making our program more extensible when making new loaded patient controllers.
    - Functionality
        - The MenuLoadedPatientController is a subclass of MenuController.
        - All controllers relating to the menu screens that appear after loading in patients inherit all the commands from the MenuController.
- **[Refactor] Used JUnit before class in the tests to reduce initialization code.**
    - Description
        - Tests were using the same code for basic initialization in each file so we decided to use the @Before decorator which reduced the length of each test file greatly.
    - Decision and Improvements
        - Initialize test variables in the before() method so we can use them in each test without the need of initializing them for every test.
        - Eliminates duplicate code code smell.
    - Functionality
        - Using the @Before decorator has the exact same functionality as copy and pasting the initialization of each test but uses a lot less code.
- **[Refactor] Aliased unload patient and sign out commands with ‘back’ command in each controller.**
    - Description
        - We wanted a uniform way of transitioning to the previous state of the program from the current state and thus the back command was born.
    - Decision and Improvements
        - Creating an alias for ‘unload patient’ and ‘sign out’ with the ‘back’ command makes our code cleaner.
    - Functionality
        - When a patient is loaded or when a user is signed in, the ‘back’ command has the same functionality as unloading the user or signing out respectively.
- **[Refactor] Added the ability to enter a number to choose a menu option.**
    - Description
        - Instead of typing out the entire command, we added the ability for users to enter the number of the command, which has the same effect as typing the command out.

    - Decision and Improvements
        - We wanted to increase the accessibility and ease of use of our program.
        - This feature satisfies the Flexibility in Use Principle of universal design.
    - Functionality
        - The number associated with a command can be seen by calling ‘help’.
- **[Refactor] Standardized date/time objects.**
    - Description
        - Standardized date/time objects to only have LocalDate and LocalDateTime objects whereas before, we had ZonedDate and ZonedDateTime objects.
    - Decision and Improvements
        - A consistent date/time format prevents the need to convert the format throughout the program, making the program more extensible and easier to maintain.
    - Functionality
        - Gets rid of the need of a time zone as we currently only have one clinic in the system and ideally all users of the program will use the same time zone as the clinic.