Part 4

1. What features were implemented?

The following features were implemented in our project: the ability to save projects (USR-01), the ability to search for projects (USR-02), the ability to add URL to favorites (USR-03), the ability to filter search results (USR-04), and the ability to get a random project (USR-05). The following functional requirements were implemented: provide a link to each repository (FR-01), initiate database entries (FR-02), the ability to login (FR-03), and the ability to search database (FR-04). The following nonfunctional requirements were implemented: provide a consistent user experience (NFR-01), ability to reset (NFR-02), ensure reliability (NFR-03) and security measures (NFR-04).

2. Which features were not implemented from Part 2?

We were unable to implement our admin user, and all of the requirement associated (USR-06, USR-07, USR -08). This was because of time restraints, and the need to develop a whole new set of features that were not important to the core functionality of the project. In addition, we removed a few bloaty parts of the search, such as number of commits and date created, as they were not part of core search function. Login validation through Google was not implemented, as we decided only to use Github as a valid form of validation.

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.

We lost all objects and variables associated with our admin requirements since they ended up not being included in the final prototype. Also, several fields were removed that were not included in the final implementation, such as number of commits and date created. See both Part 2 class diagram and final class diagram below.

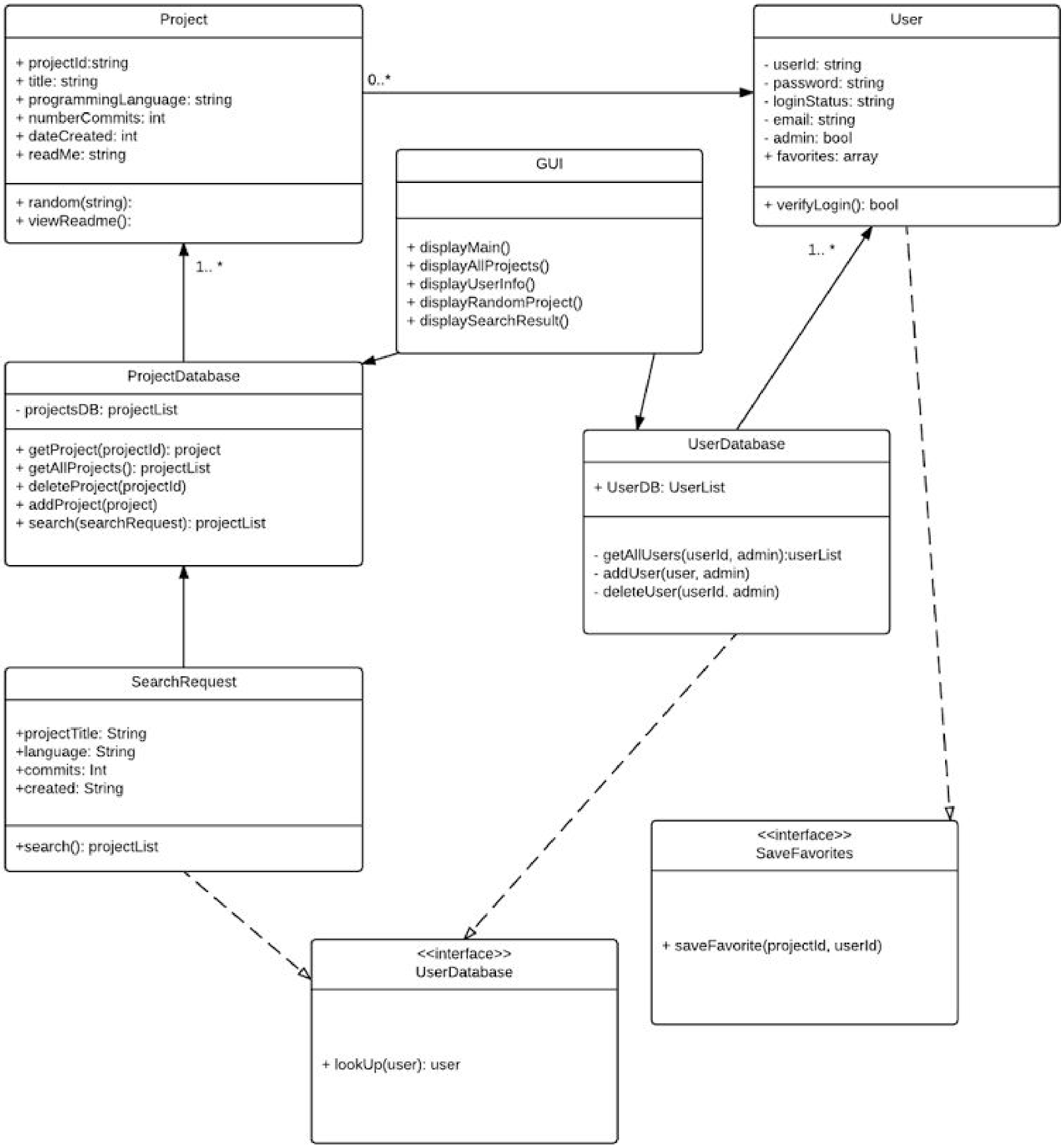
4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

Yes, we used factory design pattern. However, we could have used iterator and strategy as well. We implemented a factory to display the project info page after a search has been completed or the random button has been clicked. We could have used a Iterator to retrieve data from the mysql database. Strategy could have been used in our search functionality to select developer or language to search on for projects.

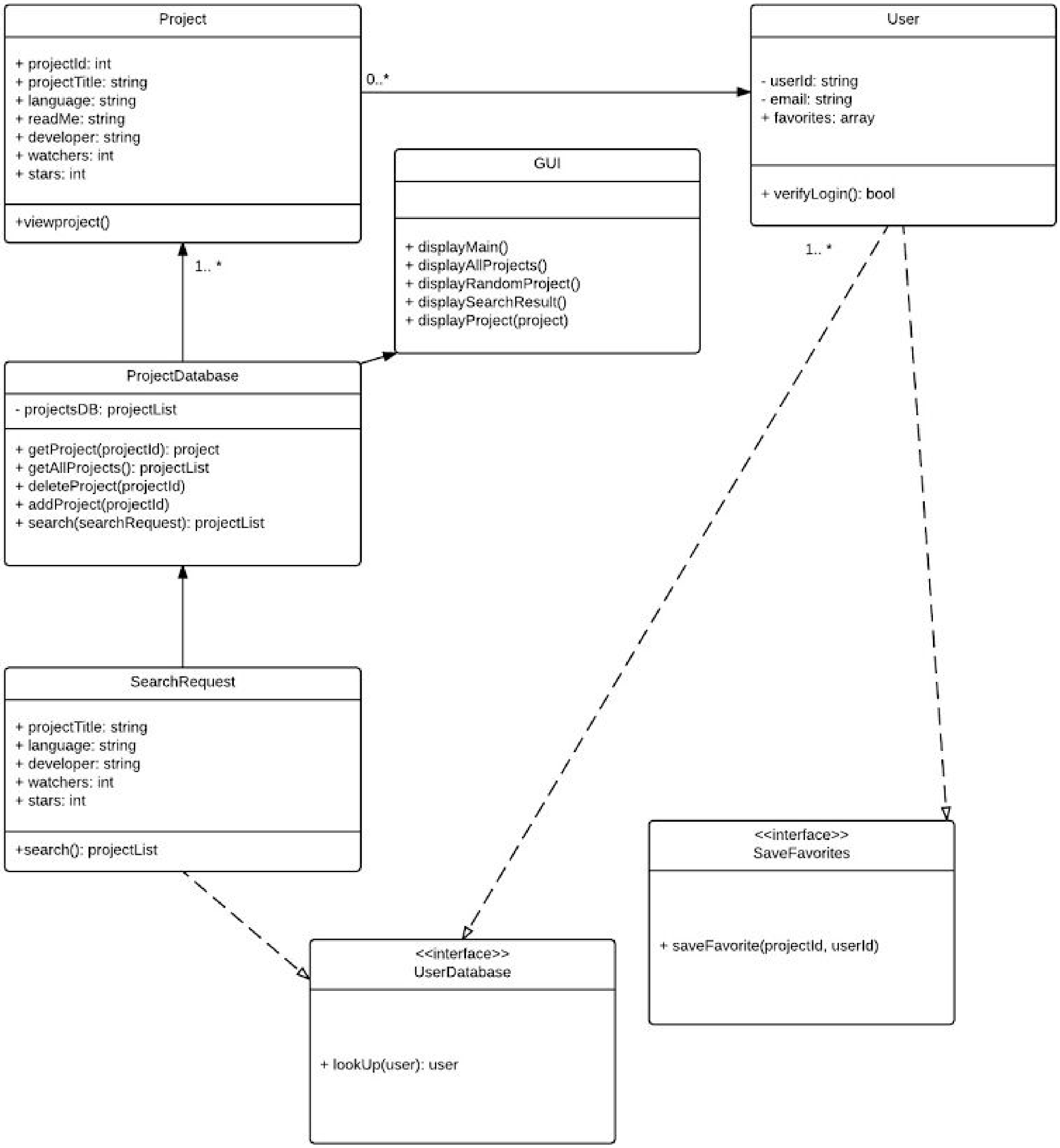
5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

Putting effort into design can really save time and effort when implementing a system, especially when it comes to flushing out how complex you can make a system in a given amount of time (saves you from making promises you can’t keep). On the other hand if you are working with technologies that you don’t have much experience with then it can slow you down, because assumptions that you make about the frameworks you are working with might not be true.

Part 2 Class Diagram



Final Class Diagram


Project
+ projectId: int
+ projectTitle: string
+ language: string
+ readMe: string
+ developer: string
+ watchers: int
+ stars: int
+viewproject()
0..*
User
- userId: string
- email: string
+ favorites: array
+ verifyLogin(): bool
GUI
+ displayMain()
+ displayAllProjects()
+ displayRandomProject()
+ displaySearchResult()
+ displayProject(project)
1.. *
1.. *
ProjectDatabase
- projectsDB: projectList
+ getProject(projectId): project
+ getAllProjects(): projectList
+ deleteProject(projectId)
+ addProject(projectId)
+ search(searchRequest): projectList
SearchRequest
+ projectTitle: string
+ language: string
+ developer: string
+ watchers: int
+ stars: int
+search(): projectList
<<interface>>
SaveFavorites
+ saveFavorite(projectId, userId)
<<interface>>
UserDatabase
+ lookUp(user): user