Kyle Nguyen, Anna Tran, Balraj Kalathil, Christopher Pallari, Elijah DelaCruz, Nicholas Mazzella, Prabhash Venkat Paila, Sidharth Lal

Team Happy

CSC 131-01

Professor Chidella

Final Deliverable2: Addressbook

This document serves as an outline and overview on the semester project created in CSC131 (Computer Software Engineering) at Sacramento State University. In the course of the Spring 2023 semester, students were tasked with creating a unique project that would satisfy Agile and Scrum project management and demonstrate competency in applying software engineering principles.

IMPORTANT RESOURCES TO REFERENCE:

Professor Chidella, the TA's and all members of Team Happy have been granted access to these.

- Github: https://github.com/csc-131-happy/Address-Database-CSC-131-
  - Unfortunately, due to the sensitivity of Planetscale's database, the github has to be set private so there are no security leaks to the public.
- JIRA: https://kyletnguyen.atlassian.net/jira/software/projects/CP/boards/1/
- Database (Planetscale): https://app.planetscale.com/csc131

Project Management/Scrum

Scrum Master: Balraj Kalathil

Product Owner: Kyle Nguyen

Both the main scrum master and main product owner are listed above. However, roles were switched weekly among team members for each member to gain experience in that role.

Sprint Planning Meeting

Team Happy planned to complete the project over the course of three sprints:

- Sprint 1 (3/15/23 - 3/22/23)
- Sprint 2: (3/30/23 - 4/12/23)
- Sprint 3: (4/13/23 - 5/12/23)

Sprint Reviews

Sprint reviews were conducted after the end of each sprint via discord or at the library. The development team, in this case, Team Happy, presented our work. Because there was no client, the team had to simulate a meeting as if there was a real client present. The team demonstrated what had been completed during the sprint and discussed the progress.

Sprint Retrospectives

Sprint retrospectives were conducted after the end of each sprint via discord or at the library. The development team reflected on the finished sprint and identified mistakes, completed goals, and achievements. Challenges and potential changes were discussed in detail. Feedback was given to each team member in order to increase productivity within the team.

Handling User Stories

In order to handle user stories and closely follow Scrum requirements, Team Happy utilized JIRA.

Two Examples of CCC (Card, Confirmation, and Confirmation)

Example 1:

Card: As a user, I want to be able to add multiple phone numbers and email addresses for a contact in my address book.

Conversation: The team discusses how this feature should be implemented and what user interface elements are needed to allow users to add and manage multiple phone numbers and email addresses.

Confirmation: The team agrees on a design for the user interface and creates a user story with acceptance criteria that cover the ability to add and manage multiple phone numbers and email addresses for a contact.

Example 2:

Card: As a user, I want to be able to search my address book by contact name or phone number.

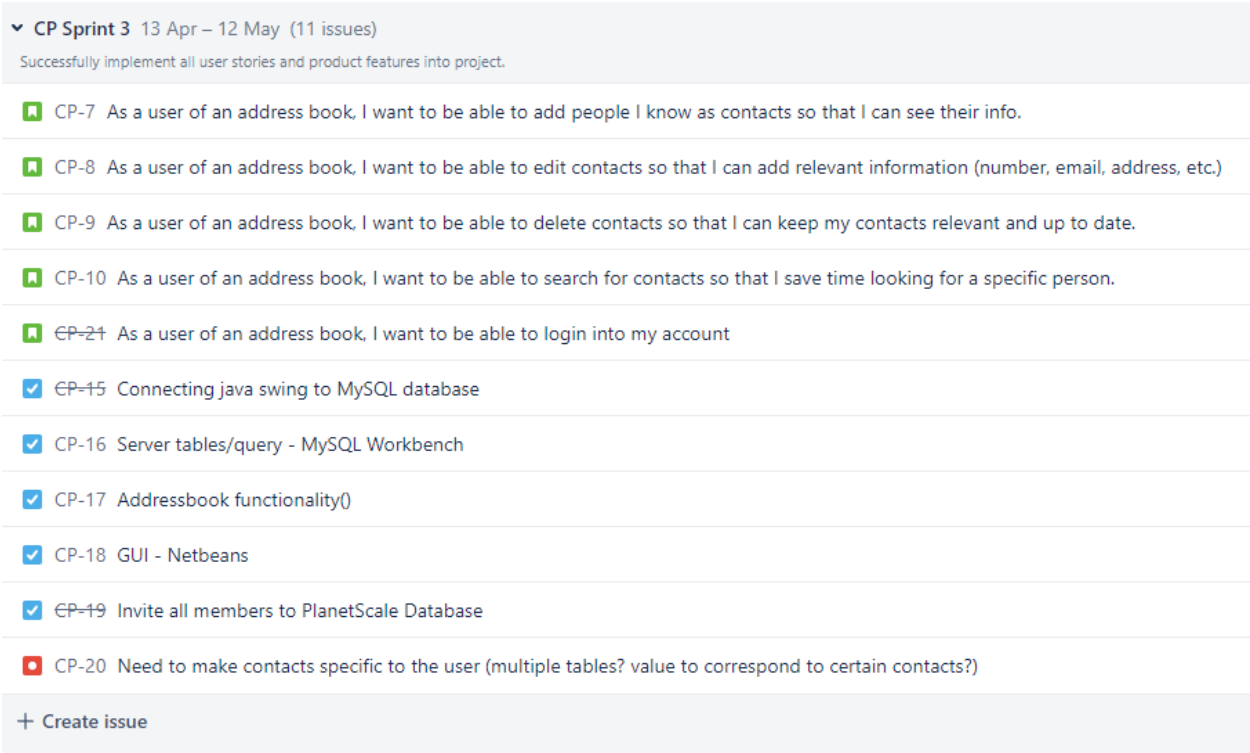
Conversation: The team discusses how the search feature should work and what user interface elements are needed to allow users to search their address book.

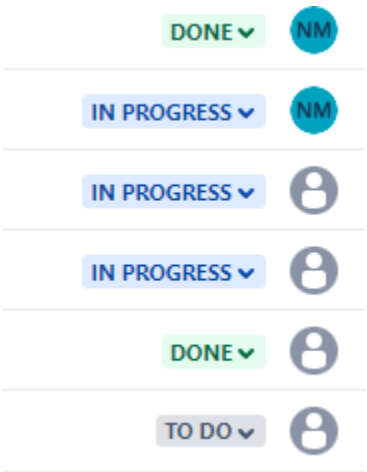
Confirmation: The team agrees on a design for the search feature and creates a user story with acceptance criteria that cover the ability to search by contact name or phone number and display the search results in a user-friendly way.

Story points, prioritization and task assignment using tool

To accomplish the implementation of user stories, prioritization, and task assignment, JIRA was used. Our Scrum Product backlog was also put onto JIRA.

Example Photos:





Risk Table

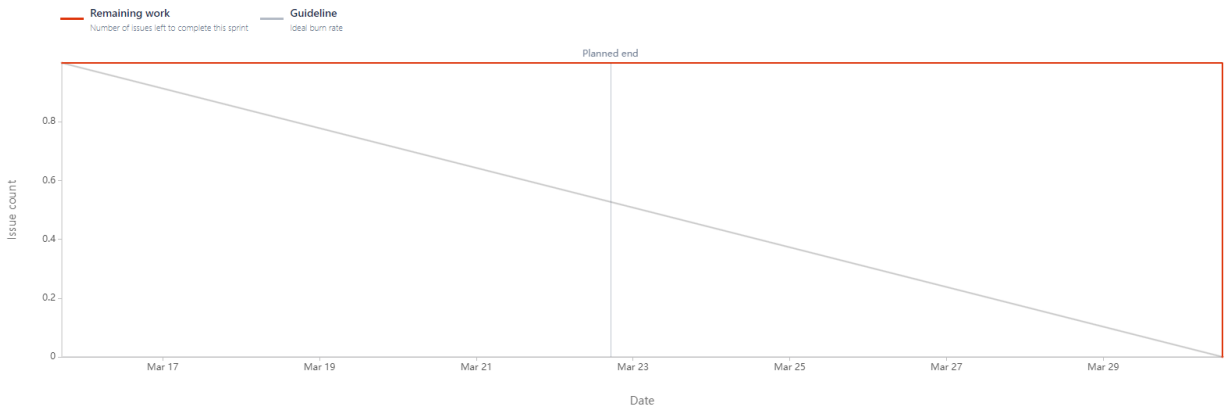
Category	Predictable Risks	Unpredictable Risks	Solutions
People	Team member illness	-Team member turnover -Failure in communication	Establish daily communication with the team and make clear documents to outline tasks.
Process	Failure or incompetency in following Scrum	-Technology failure -Unanticipated client requirements or changes	Daily check-ins with clients and constant communication with Scrum master. Following requirements closely.
Product	Incomplete Features	-Security breaches -Unexpected collapse of supporting website for database	Conduct regular security testing and have thorough database passwords. Have a backup plan for failures.

Sprint Burndown Chart

Sprint 1:

Date - March 15th, 2023 - March 22nd, 2023

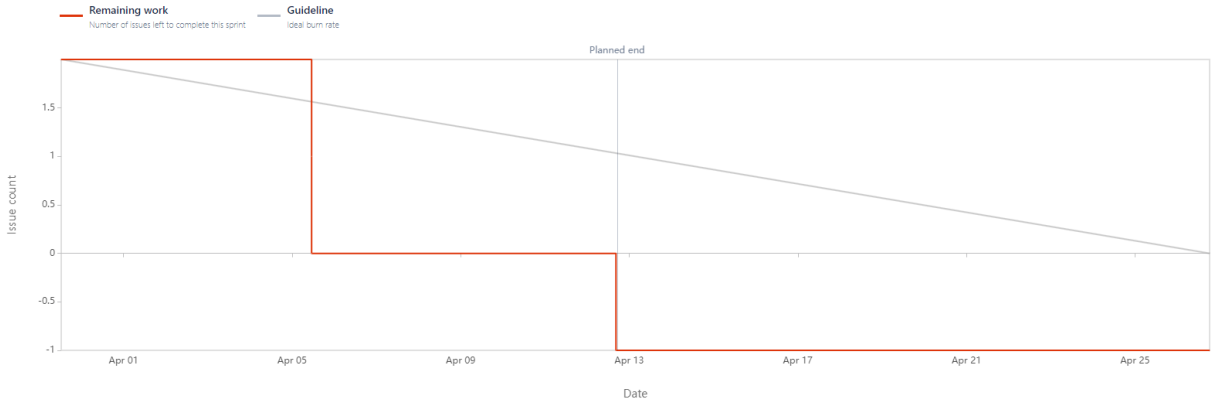
Sprint goal - To get all team members of Happy in CSC131-01 to set up a JIRA account and join the project.



Sprint 2:

Date - March 30th, 2023 - April 12th, 2023

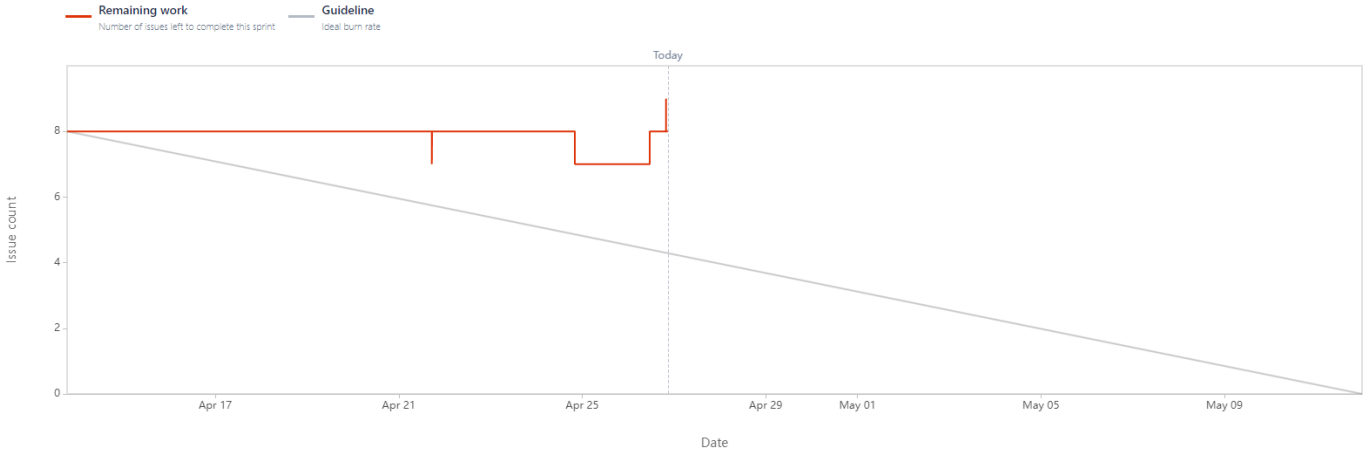
Sprint goal - Create Java Project in Eclipse and connect to JIRA, Github, and MySQL (cloud based host).



Sprint 3:

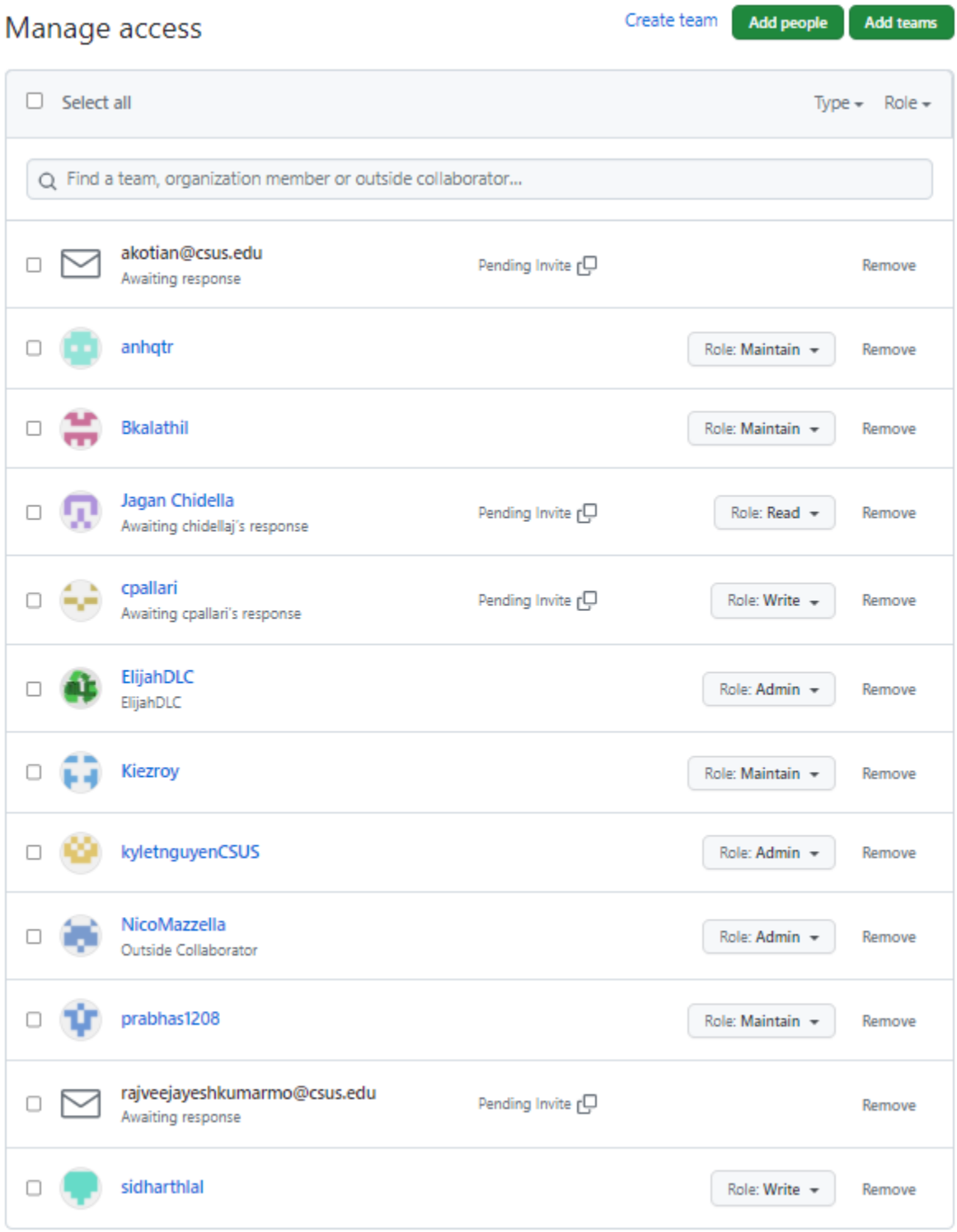
Date - April 13th, 2023 - May 12th, 2023

Sprint goal - Successfully implement all user stories and product features into project.



Coding Using Github

Proof of Collaboration: https://github.com/csc-131-happy/Address-Database-CSC-131-



Written description of High Cohesion and Low Coupling principles

In the context of our address book project, high cohesion would mean all the related code is close together in order to achieve a single task or functionality. The components should be self contained and highly focused. Low coupling would refer to the interdependence between the components with loose connections and minimal dependencies.

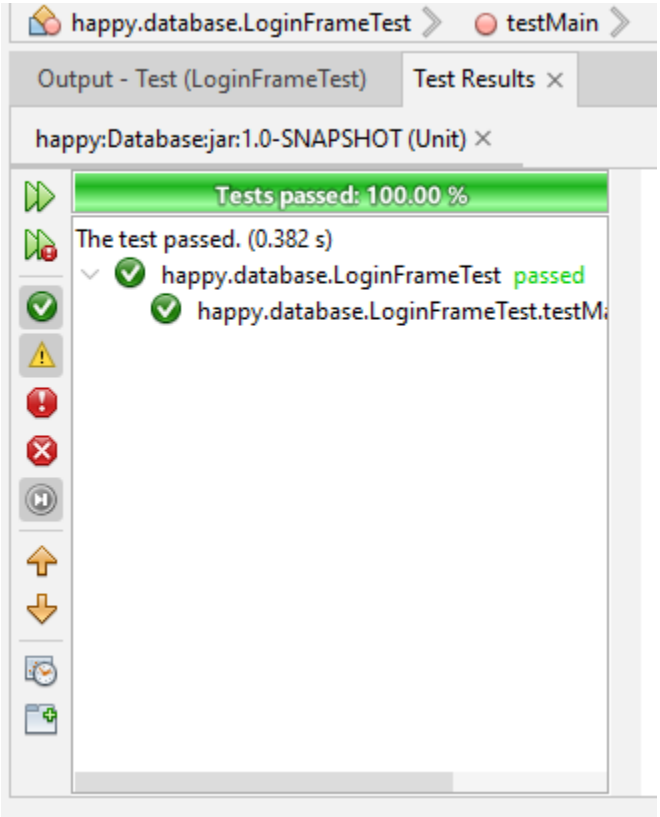
Example of High Cohesion:

JFrameAddressBook.java: This class is a representation of high cohesion because it is solely responsible for managing the address book data. This includes but is not limited to: adding, deleting, and updating contacts. It is not related to presentation or any GUI (Graphical User Interface) details.

Example of Low Coupling:

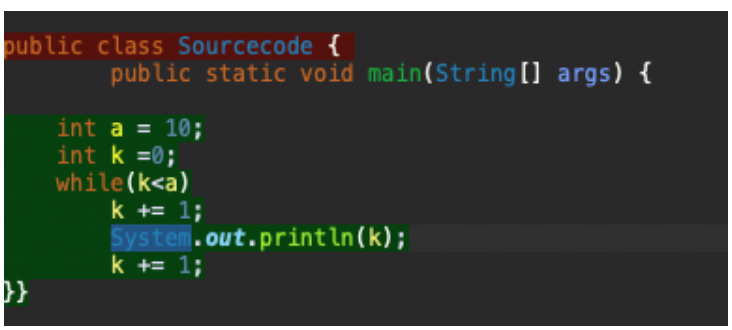
Team Happy addressed the principle of low coupling through creating multiple classes to separate concerns. We created separate and independent classes for creating, deleting, and editing contacts. This way, each class had a specific responsibility and had no effect on other classes with the project.

JUnit



Test case would pass if the application was able to successfully connect to the database and launch.

EclEmma



```
public class Sourcecode {
        public static void main(String[] args) {

    int a = 10;
    int k =0;
    while(k<a)
        k += 1;
        System.out.println(k);
        k += 1;
}}
```

The part highlighted in green represents the code that is executed. Yellow part represents partially executed code and the red part represents the code that is not executed.

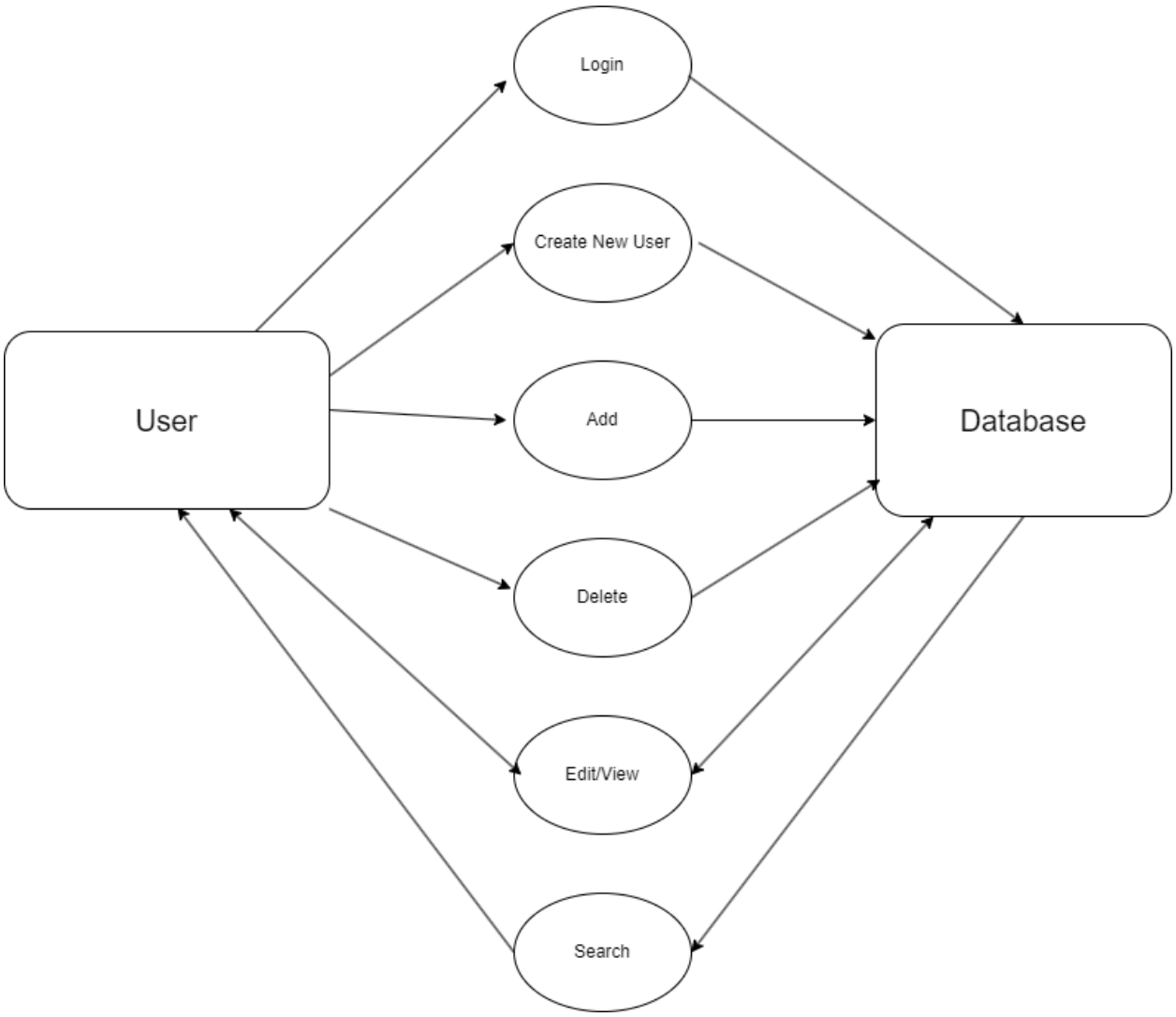
TestCoverage	100.0 %	1	0	1
src	100.0 %	1	0	1
(default package)	100.0 %	1	0	1
Sourcecode.java	100.0 %	1	0	1

This is the detailed coverage report of the file.

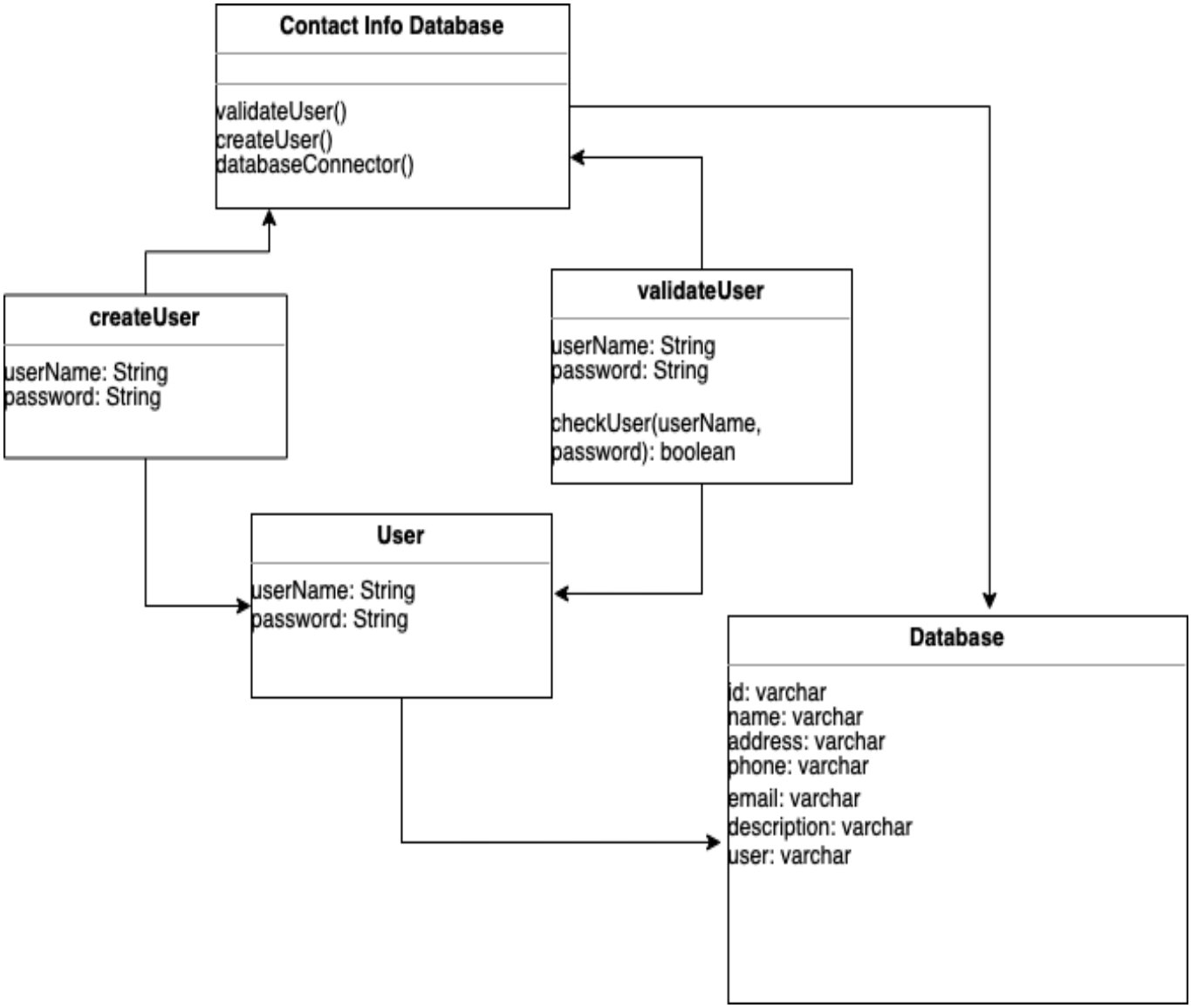
Four Types of UML Diagrams

NOTE: Due to the use of multiple symbols, it was not possible to use LucidChart to create the diagrams without paying a fee. However, the alternative of Draw.io was used.

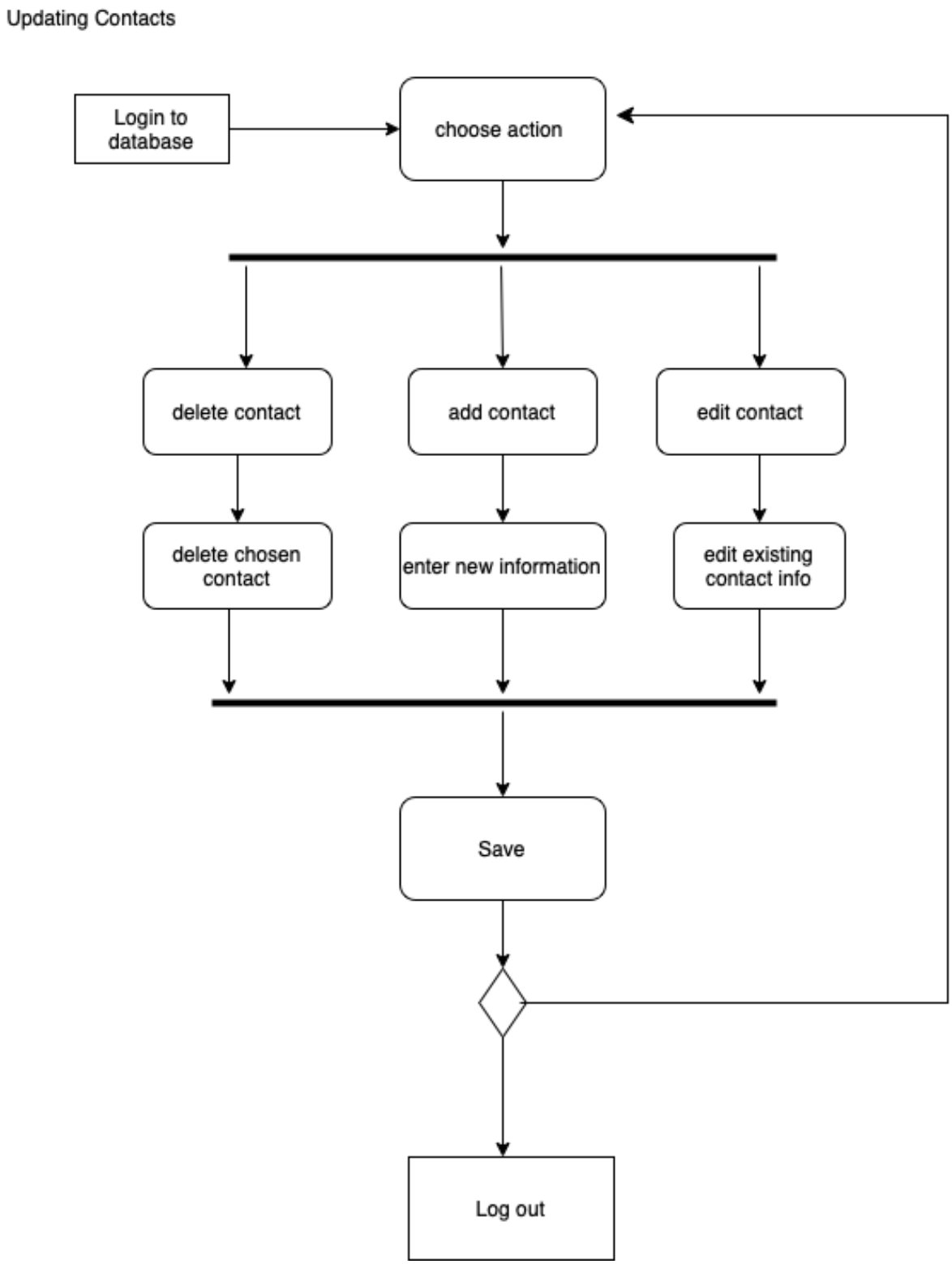
Use Case Diagram



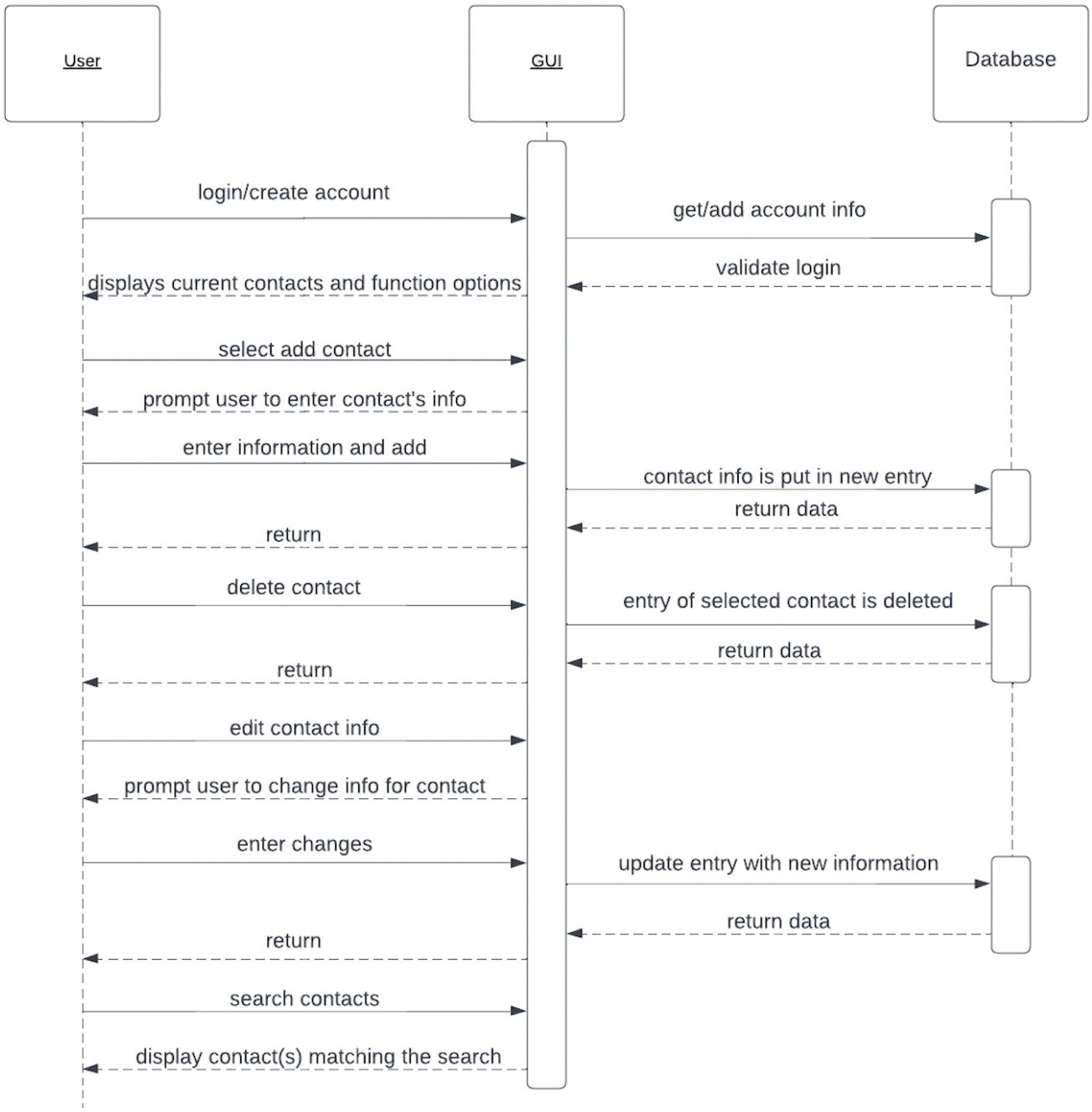
Class Diagram



Activity Diagram



Sequence Diagram



Minimum 2 Design Patterns

Design A (Factory)

The first design pattern that Team Happy implemented in the AddressBook Project was the Creational Design Pattern, specifically a Factory pattern. This can be seen within the NewUserFrame.java class. This class is responsible for the creation of GUI components and objects which demonstrates the concept of initializing and creating objects. It creates new instances of the LoginFrame when the “Cancel” button is activated or a new user is created.

Design B (Composite)

The next design pattern that Team Happy implemented in the AddressBook Project was the Structural Design Pattern, specifically a Composite Pattern. The class that best follows this pattern is the ViewFrame.java class. The code here is focused on organizing the GUI components and their layout.

Bonus Points

1a.

User Story: As a user, I want to be able to search for contacts by name, phone number, or email address so that I can easily find the contact information I need.

INVEST Principles:

- Independent: This user story is independent of any other user stories in the project.
- Negotiable: The details of the search functionality can be negotiated with the development team.
- Valuable: This user story provides value to the user by making it easier to find contact information.
- Estimable: The development team can estimate the effort required to implement this user story.
- Small: This user story is small enough to be completed within one sprint.
- Testable: The success of this user story can be tested by verifying that the search functionality works correctly.

MOSCOW Principles:

- Must have: The search functionality must be able to search by name, phone number, and email address.
- Should have: The search results should be displayed in a clear and organized way.
- Could have: The search functionality could have the ability to filter results by additional criteria such as location or job title.
- Won't have: The search functionality won't have the ability to search by custom fields.

1b.

User Story Map:

Epic: Address Book Application

User Story 1: Search for Contact

User Story 2: Add Contact

User Story 3: Edit Contact

User Story 4: Delete Contact

User Story 5: Login and logout with unique credentials

1c.

Definition of Done:

- The search functionality allows the user to search for contacts by name, phone number, or email address.
- The add functionality allows the user to add a new contact that is assigned to their unique address book
- The edit functionality allows the user to edit an existing contact in their address book and save the changes
- The delete functionality allows the user to delete an existing contact
- Logging in and logging out works as expected with correct credentials

2.

Observer Pattern: This pattern was used in the project to keep track of the changes in the address book and notify the correct parties (user interface components, etc.) of the changes.

Strategy Pattern: This pattern was used in the project to implement different sorting and filtering algorithms within the address book and allows the user to choose which method they prefer.

3.

Java Swing was used for the interaction design pattern and serves a purpose of creating user interface.