CS 2340: Objects and Design - Final Exam

TEAM NAME/NUMBER: _____Group 47_____

The following team members participated in the final exam (do not list any team member who did not participate in completing this exam):

Printed Team Member Names
Nandini Ramakrishnan
Jon Wang
Ananth Vivekanand
Vidit Pokharna
Dhruv Adha
Shivansh Sharma

Questions

1. (20 pts) Requirements: For the usage scenarios provided in the write-up, complete the correct number of use cases for each team member participating in this final exam given below. (Nandini Ramakrishnan)

- Number of use cases:
  - 4+ person team: 1 use case per person

- Nandini Ramakrishnan: As an unregistered student, I want to create an account and login so that I can track my grades.
  - **Given** the system user is an unregistered student,
  - **When** the student creates an account and logs in
  - **Then** they should be able to track their grades.
- Jon Wang: As an unregistered teacher, I want to create an account and login so that I can input grades for my students.
  - **Given** the system user is an unregistered teacher,
  - **When** the teacher creates and account and logs in
  - **Then** they should be able to input grades for their student.
- Ananth Vivekanand: As a registered student, I would like to be able to view the grades of previously enrolled courses so that I can get a breakdown of how my final grade was calculated.
  - **Given** the system user is a registered student,
  - **When** the student views the grades of previously enrolled courses
  - **Then** they can get a breakdown of how the final grade was calculated.
- Vidit Pokharna: As a registered parent, I want to be able to view my child's grades and feedback so that I can monitor their progress in school.
  - **Given** the system user is a registered parent,
  - **When** the parent is able to view their child's grades and feedback
  - **Then** they can monitor their progress in school.
- Dhruv Adha: As a registered teacher, I want to be able to enter grades for students so that students can monitor their academic performance.
  - **Given** the system user is a registered teacher,
  - **When** the teacher enters grades for students
  - **Then** the students can monitor their academic performance.
- Shivansh Sharma: As a registered student, I would like to be able to view my schedule so that I know where to go at any given time.
  - **Given** the system user is a registered student,
  - **When** the student views their schedule
  - **Then** they would know where to go at any given time.

- You must have at least 3 actors.

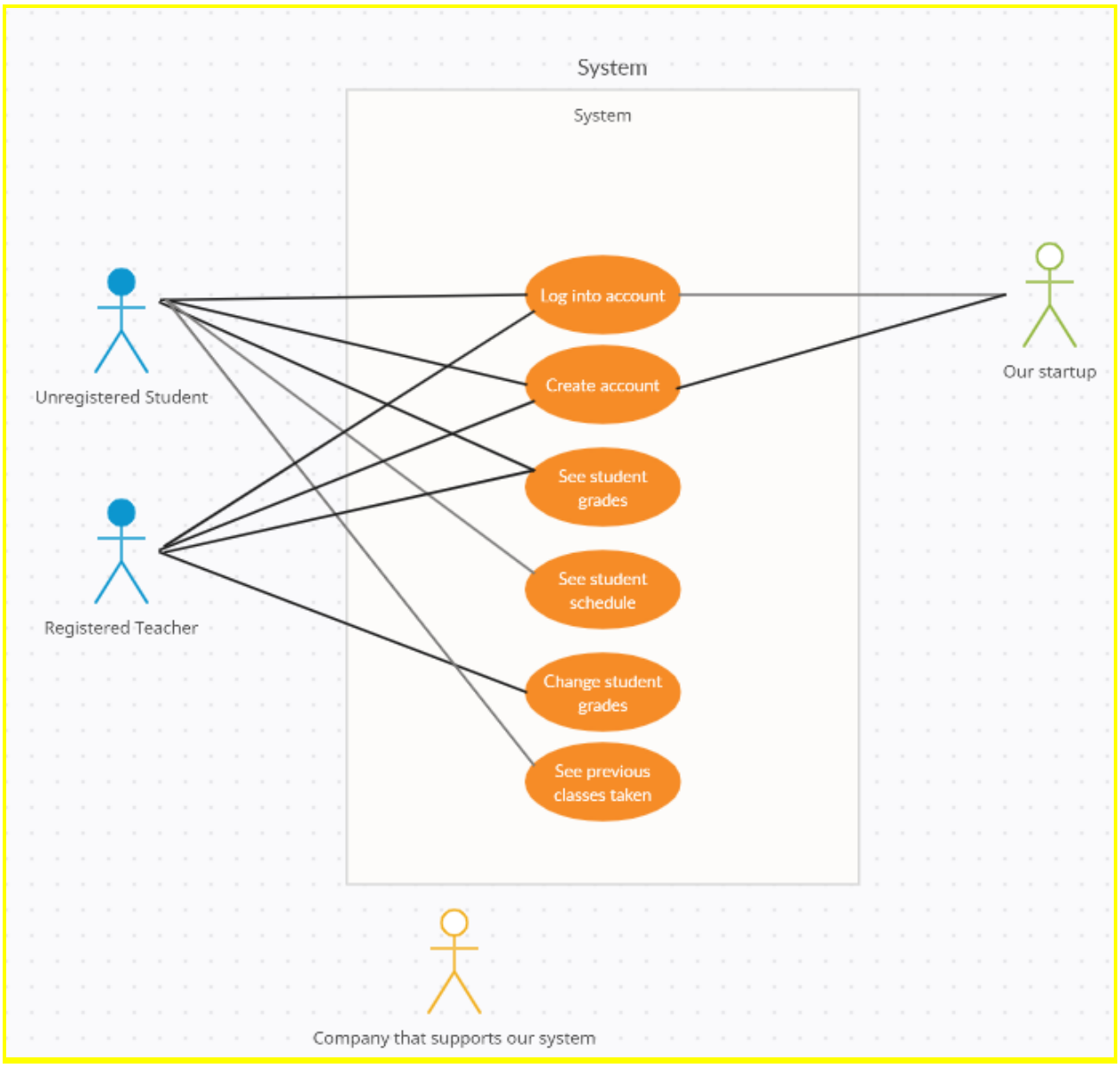
- **Actors**
  - **Primary Actors**
    - Unregistered student: a student that has not yet enrolled, but can create and account when they want to start taking classes. They use the system to do so.
    - Registered student: a current student of the school who uses the system to track their grades, see how their final grades were calculated, and view their schedule
  - **Secondary Actors**
    - Our startup: the company that redesigns the school district's online information system
  - **Off-stage Actors**
    - Company that supports our system: the company that lets people access the system, whether that be the Google Play Store, an App store, or another company

- The format must be casual or fully dressed to be graded (not brief).
- You must include the main success scenario along with at least 1 alternate scenario.

- **Main Success Scenario**
  - The students, both registered and unregistered, are able to log into the accounts they've created, see their current grades, past grades, their transcripts, and their schedules. On the other hand, registered and unregistered teachers can create accounts and log in to input grades for their students, and parents are able to monitor their children's progress in school by seeing their grades when they log into the system.
- **Alternate Scenarios**
  - Registered students' accounts have been deleted, and they are not able to see their past courses anymore.
  - The system blocks parents from seeing their children's grades.
  - Teachers are not able to change the grades they give their students, even if there was a mistake initially.

- +5 extra pts if you include a Use Case Diagram that shows your actors and use cases.

Use Case Diagram:


System
System
Log into account
Create account
See student grades
See student schedule
Change student grades
See previous classes taken
Unregistered Student
Registered Teacher
Our startup
Company that supports our system

2. (10 Points) Analysis: Perform a basic OO business analysis (OOA) of the application described in the problem document and outlined in your use cases. You must produce a domain model of the problem given. However, you may take other steps as needed.
(Vidit Pokharna + Jon Wang)

Classes:

- Student
- Parent
- Teacher
- Grade
- Schedule
- Course Content
- Assignment
- Feedback

Attributes:

- Student: ID, Name
- Parent: ID, Name
- Teacher: ID, Name
- Grade: Value, Course ID, Student ID, Teacher ID
- Schedule: ID, Day, Time, Course ID
- Course Content: ID, Title, Description
- Assignment: ID, Title, Description, Due Date, Course ID, Student ID
- Feedback: ID, Description, Student ID, Teacher ID

Relationships:

- A Student has one or more Grades.
- A Student has one or more Assignments.
- A Parent can view the Grades and Assignments of their child (Student).
- A Teacher can input Grades, provide Feedback, and create Course Content.
- A Course Content can have many Assignments.
- A Course Content can have many Schedule.
- A Schedule can be assigned to one or more Courses.

User Stories:

1. As an unregistered student, I want to create an account and login so that I can track my grades.
2. As an unregistered parent, I want to create an account and login to be able to access my child's academic information without having to rely on my child to provide their login credentials, so that I can easily monitor their academic progress and provide support as needed.
3. As an unregistered teacher, I want to create an account and login so that I can input grades for my students.
4. As a registered student, I would like to be able to view the grades of previously enrolled courses so that I can get a breakdown of how my final grade was calculated.
5. As a registered parent, I want to be able to view my child's grades and feedback so that I can monitor their progress in school.
6. As a registered teacher, I want to be able to enter grades for students so that students can monitor their academic performance.
7. As a registered teacher, I want to leave feedback about a student's performance and behavior so that the parent is aware of how their child is doing.
8. As a registered teacher, I want to be able to import my grades through a 3rd party API so that grades can be published for externally reviewed assignments.
9. As a registered student, I would like to be able to view my schedule so that I know where to go at any given time.
10. As a registered parent, I would like to be able to view the schedule of my child so that I am informed about their daily academic activities.
11. As a registered student, I would like to be able to view and download class materials, such as lecture slides and homework assignments, so that I can study and complete assignments outside of class.
12. As a registered student, I want to turn in an assignment so that it can be graded by my teacher to assess my work.
13. As a registered parent, I want to be able to monitor my child's submitted assignments and communicate with their teachers through the system so that I can stay informed and support my child's education.
14. As a registered teacher, I want to be able to create and organize course content and assign homework so that I can streamline my workflow and provide more timely feedback to students.

Domain Model:

- Account
  - username: String
  - password: String
  - id: int
- StudentAccount
  - name: String
  - parent: Parent
  - schedule: Schedule
  - feedbacks: List<Feedback>
  - age: int
  - eligible: bool
- TeacherAccount
  - name: String
  - courses: list<Course>
- ParentAccount
  - name: String
  - students: list<Student>
- Course
  - name: String
  - description: String
  - teacher: Teacher
  - assignments: list<Assignment>
- Assignment
  - name: String
  - submitted: bool
  - grade: Grade
- Grade
  - value: float
  - feedback: Feedback
- Feedback
  - value: String
  - course: Course
- Schedule
  - courses: list<Course>

Domain Model Diagram:
https://drive.google.com/file/d/1wsoX3v8Jor_9eRDgR8Ce1glZiYMCX1Y_/view?usp=sharing

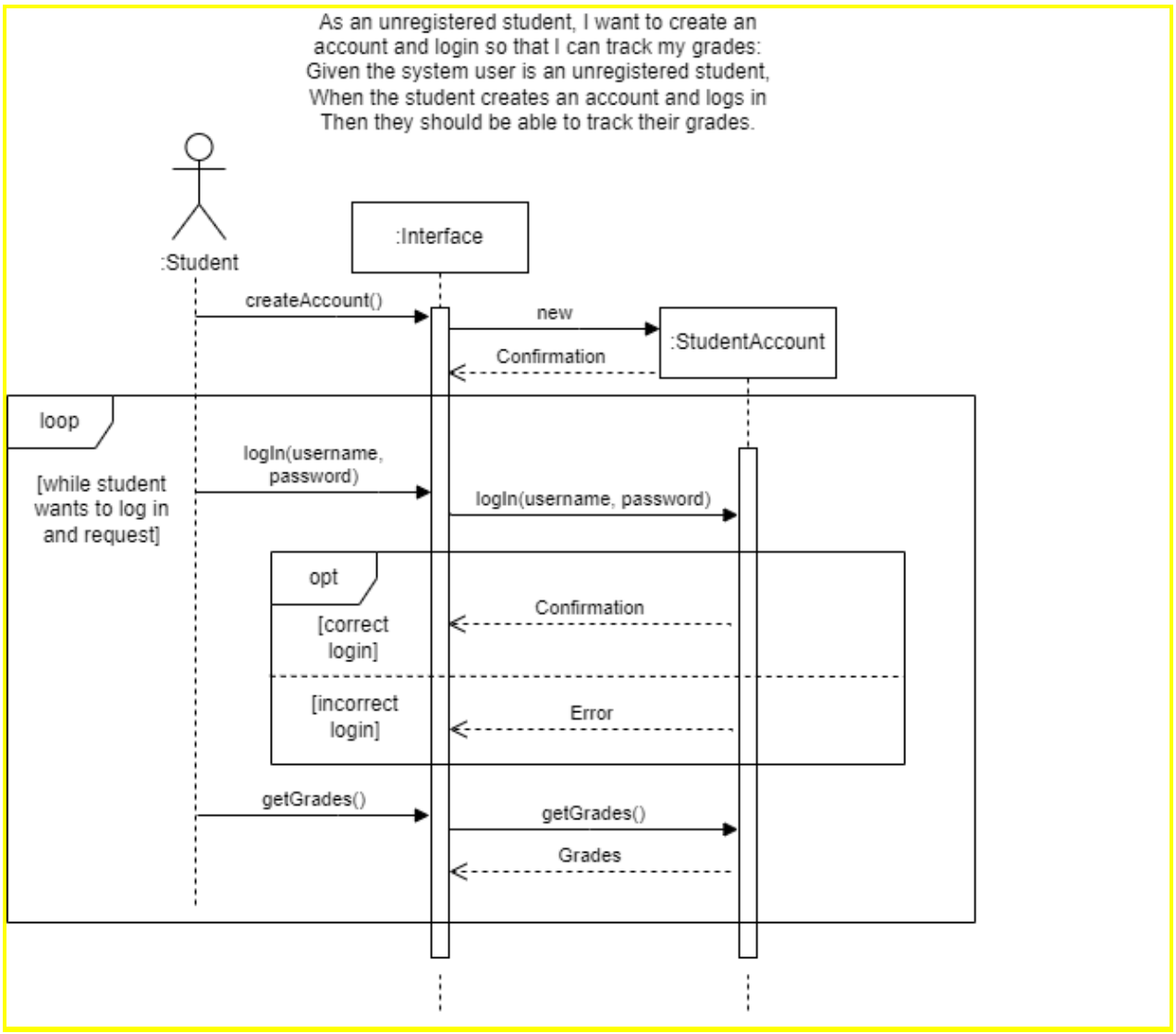
3. (15 Points) Design #1: Create a UML class diagram for the application described in the problem document. You should model the basic domain (back-end) classes in the application. Ensure that your diagram has the following constructs:

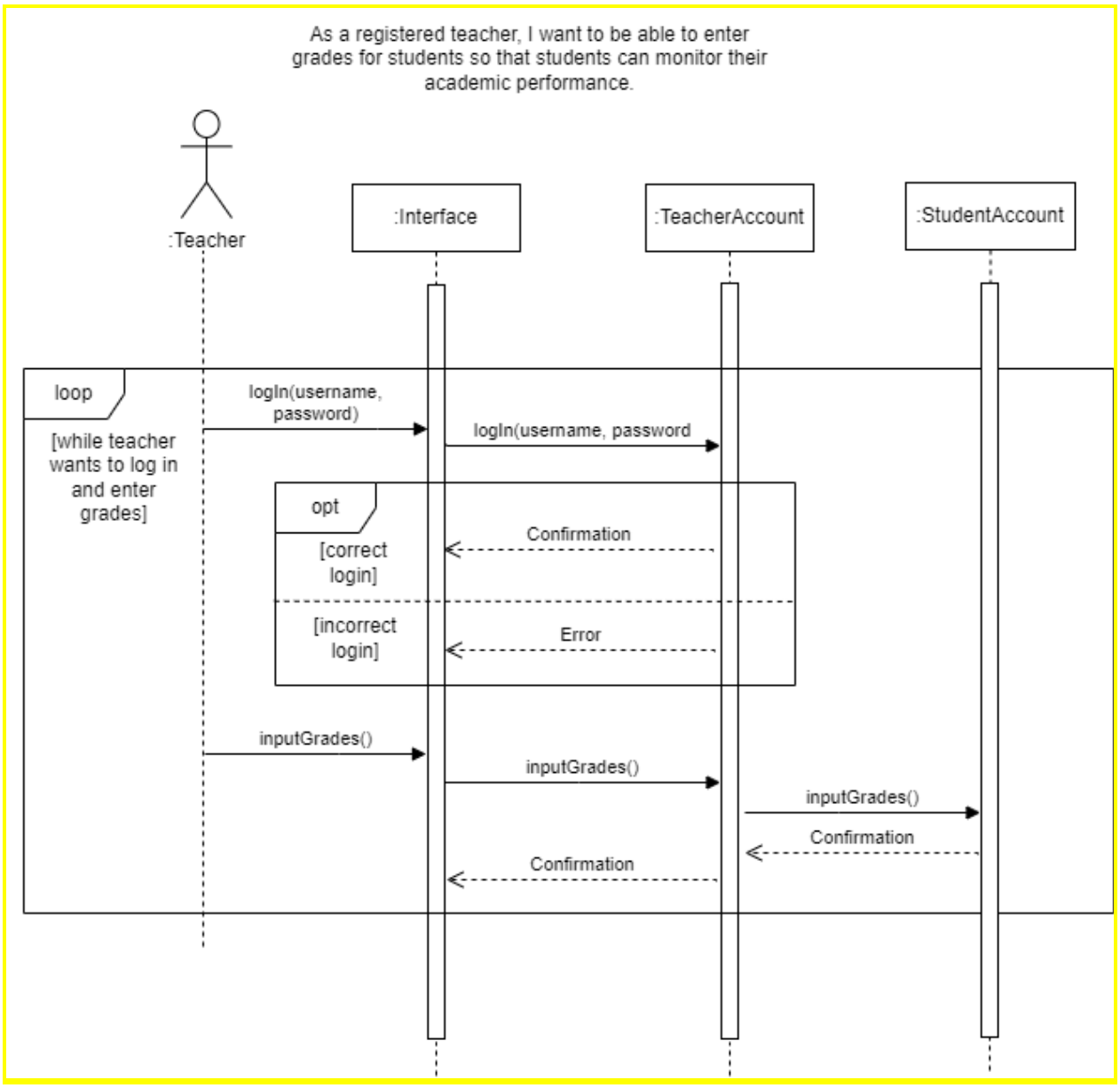
- Inheritance, an Interface, an abstract class, a class that implements your interface
- 4 Instance Variables at least one protected, one privately and one publicly accessible
- 4 Instance Methods at least one of which takes parameters, one which has no parameters, one that returns something, and one which returns nothing (a method may satisfy more than one of these if desired). Getters and setters do not count toward this requirement and should be excluded.
- You must also show appropriate multiplicity constraints and associations with correct UML notations.

*** You do NOT need to model any networking/communication and UI (View) classes. If you need such classes for a later question, you may abstract the entire UI into a single class.
(Shivansh Sharma + Dhruv Adha)

https://drive.google.com/file/d/1LXGdtTEF6aZZCh13Jrefyyh5SPI4J0v5/view?usp=sharing

4. (10 Points) Design #2: For a team of 5 or more, for 2 of the use cases you developed above, draw UML sequence diagrams. For a team of 4 or fewer, for 1 of the use cases you developed above, draw a UML sequence diagram. Be sure each diagram is consistent with the UML class diagram and the use case you have previously completed (upholds conceptual integrity). Failure to uphold conceptual integrity may result in a deduction of points.
(Shivansh Sharma + Dhruv Adha)




As a registered teacher, I want to be able to enter grades for students so that students can monitor their academic performance.
:Teacher
:Interface
:TeacherAccount
:StudentAccount
loop
[while teacher wants to log in and enter grades]
logIn(username, password)
logIn(username, password
opt
[correct login]
Confirmation
[incorrect login]
Error
inputGrades()
inputGrades()
inputGrades()
Confirmation
Confirmation

5. (10 Points) Architecture: Your client forgot to tell you that the application must be multi-platform, meaning they want an iPhone, Android, and web application. How would you change your design to accommodate this requirement? Note, you can assume that the UI/front-end is one component, you don't need to design the front-end. Is there a name for this type of architecture? For a team of 2 or fewer, this question is optional.
(Jon Wang)

We would use a Model-View-ViewModel architecture to support multiple platforms. This would allow our backend, or Model, to be platform independent. We wouldn't need to change any of our design, because our design never directly interacts with the frontend. Our design simply handles events when triggered by the ViewModel and updates the ViewModel as certain properties/data change in the Model. This means all versions of our UI can call the same methods exposed by the ViewModel to interact with the Model. Thus, all we would need to do is build a UI for each platform and build the ViewModel that is used to connect the View and Model. The ViewModel would have a handler for each user interaction mentioned in part (1) as well as update methods to inform the View that a value it's bound to has changed. Note that this architecture is especially preferable, because we are also able to make the ViewModel platform independent (or at least mostly independent). This allows for maximum code reusability and makes it easier to support more frontend platforms in the future.

6. (10 Points) Design Patterns: Look at your design and UML diagrams. Describe what design pattern you would use to help integrate the API calls to external services so that the APIs could be added and removed easily. Annotate the design pattern components on your UML diagram and label them as a separate diagram.
(Jon Wang + Dhruv Adha)

We could use a combination of the Facade and Adapter patterns. The Facade would be the interface that provides a simplified view of the API return data to the client code. This helps the client adhere to the Principle of Least Knowledge. Even as the APIs change, the client remains unaffected because the behaviors exposed by the Facade stay the same. The Adapters would implement the external API connections, calls, and data parsing. The Adapters would then encapsulate or smooth over differences between the outputs from the external APIs to provide a common interface for the Facade to call. This makes it easy to add and remove external API services, because they will conform to the same Adapter interface, meaning the Facade won't have to change its behavior. To add an API, all we'd have to do would be to implement the methods required by the Adapter interface and then plug into the Facade. To remove an API, all we'd have to do would be to migrate the Facade calls to an Adapter for that API over to a different API that implements the same behavior. These design choices reduce coupling in our code, leading to easier maintenance.

7. (15 Points) Design Principles #1: You decide to use Mongo-DB for data storage. Another team in the company is going to design and develop the database separately from your team, which is implementing the core business logic for the application described in the problem document. Describe what steps you need to take to allow you to test your application without the database and to allow someone to change to MySQL in the future without affecting your main codebase (Defend your design). You may leverage design pattern(s) and/or design principles to satisfy this requirement. You do not have to modify your design/diagrams just explain how you would do so.
(Ananth Vivekanand + Jon Wang)

We would first design an interface for interacting with the database. We would use the repository pattern, because it is focused on matching our domain model. The repository pattern allows us to decouple the business logic layer and the database layer. The repository will expose methods to create, read, update, and delete domain objects from the database and abstract away the actual implementation of these queries. Thus, if we change to MySQL in the future, the business logic doesn't have to change because it was programmed to an interface. All we'd have to do is to reimplement the methods of the interface to work with MySQL. Another competing pattern would be the DAO pattern. The reason we chose the repository pattern is because it provides a higher level of abstraction and will interact more nicely with our business logic, since it's model driven (i.e. it doesn't expose as much/at all the database schema). The drawbacks are that we'd have to specially design the database schema to conform to the domain model. For example, we may need extra association tables to be able to follow the domain model. However, we decided that while this incurs more cost and adds complexity at the database layer, its benefits in terms of a more decoupled and abstracted database layer will outweigh the costs in the long run through maintenance reduction.

We could allow for business logic testing by creating a mock database that implements the interface detailed above. We could use a framework like mockito to emulate the database in software. We would create objects with certain return values and behaviors to test the business logic. This allows us to modify the objects' behavior to test various edge cases that would come up with a real database.

8. (10 Points) Design Principles #2: You are looking at some code written by a subcontractor for the application described in the problem document. Your boss wants to know if anything needs to be changed before we pay them. Discuss what you will look for to determine if the code adheres to SOLID principles.

- For a team of 5 or more, you must evaluate based on at least 3 of the SOLID principles.

(Ananth Vivekanand)

SOLID provides a set of principles to make OO designs maintainable, flexible, and understandable. One thing to look for would be adherence to the Single Responsibility Principle, which can be checked through the features of each class. Classes that seem to have multiple responsibilities should be broken up into smaller, focused interfaces that can be chosen to be implemented as needed. This reduces the necessary reasons to change a class, improving maintainability and understandability. I would look through the codebase for frequent usage of specialized interfaces that accomplish specific responsibilities.

Another aspect of SOLID I'd check for would be the Liskov Substitution Principle, in which subclasses should be able to replace usages of superclasses. I'd map out inheritance relationships between classes and check whether all necessary methods of superclasses are overridden/implemented in subclasses and whether they're on par in terms of functionality. Subclass implementations that are radically different are likely a violation of the Liskov Substitution Principle and are the wrong abstractions to have.

We could additionally inspect the dependencies between modules to see whether they're based on abstractions or details to validate adherence to the Dependency Inversion Principle. For example, if a class encapsulates another dependency, it should refer to it by an interface rather than a concrete implementation. This ensures dependencies can be changed out for each other and reduces the information that a class needs about a dependency. Where Liskov's Principle focuses on inheritance, DIP focuses on dependencies.

Extra Credit #1 (Up to 10 points)

Describe a good project that would be appropriate for 2340 next Summer or Fall. You need to supply at least 10 features to be implemented and the reason why you think it is a good project. In addition, describe how you would break the features into 4-5 iterations that are of an acceptable Agile sprint length of your choosing (make sure to tell me how long). You may pick either an Android or JavaFx application but must clearly state which you intend the project to be implemented as.

(Vidit Pokharna)

Project: An Android App for Fitness

Features:

1. User registration and login page
2. User profile creation page (taking in metrics such as height, weight, age, gender, etc.)
3. Update progress page to allow daily workouts and steps traveled
4. Fitness goal page, to set and track fitness aims for steps taken, calories burned, etc.
5. Custom workout plans that can be created based on fitness goals
6. Reminders (paired with phone or just within the app as a page)
7. Social messaging feature
8. Friends page to create workout goals, compete, etc.
9. Integration with health tracking on phone (much more difficult to add to app)
10. Generate reporting after certain duration of time

Reasons why it's a good project:

- Health and fitness are important aspects of people's lives
- Fitness tracking application can help users achieve their fitness goals.
- The fitness tracking application can be customized to fit different user needs and preferences (if properly coded)

Iterations breakdown:

- Iteration 1 (2 weeks):
  - User registration and login functionality
  - User profile creation functionality
- Iteration 2 (2 weeks):
  - Progress page functionality
  - Fitness goal page functionality
- Iteration 3 (2 weeks):
  - Custom workout page functionality
  - Progress tracking and reminders functionality
  - Social messaging feature functionality
- Iteration 4 (2 weeks):
  - Friends page functionality

    - Integration with health tracking functionality
- Iteration 5 (2 weeks):
    - Report generation and analytics functionality

Extra Credit #2 (Up to 20 points)

UI/Prototype: With respect to the application described in the final exam problem document, draw/mockup (lofi or hi-fi) a UI Flow diagram that outlines the major screens, basic widgets on each page, and how the user navigates between the screens. We did not cover this in class, but you can read more about it here: http://agilemodeling.com/artifacts/uiFlowDiagram.htm
(Nandini Ramakrishnan)