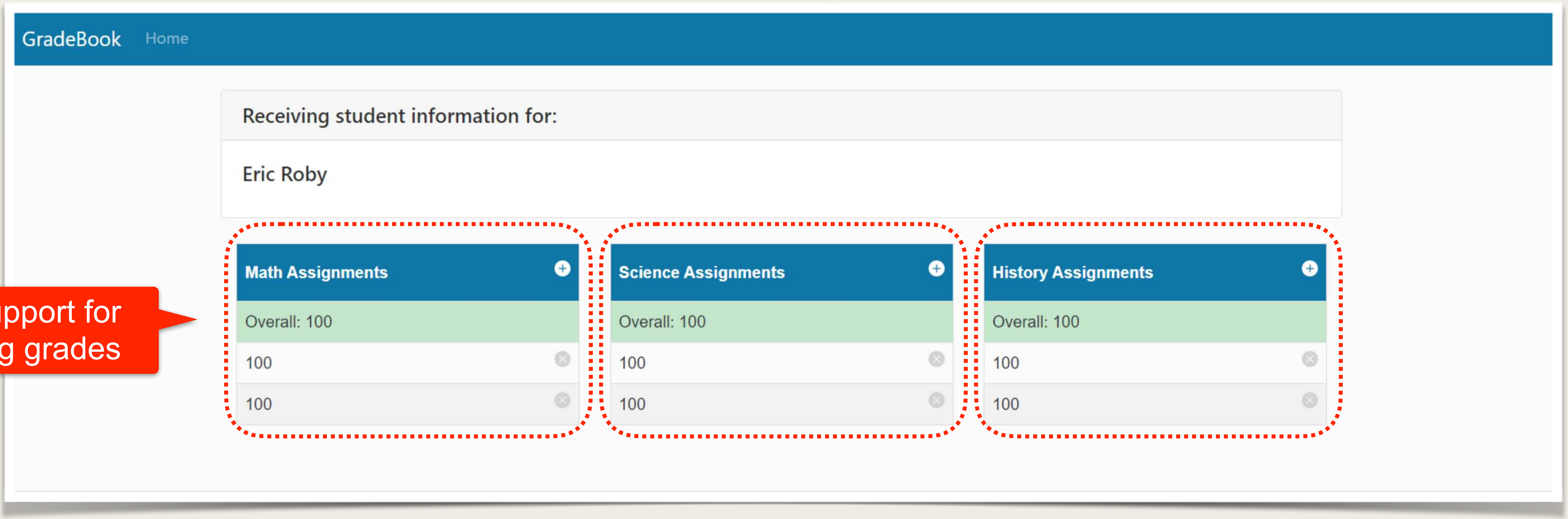
Create Grades

To Do: Grades

- Currently the app does not keep track of grades for a given student
- At the moment, the UI is hard coded



Receiving student information for:

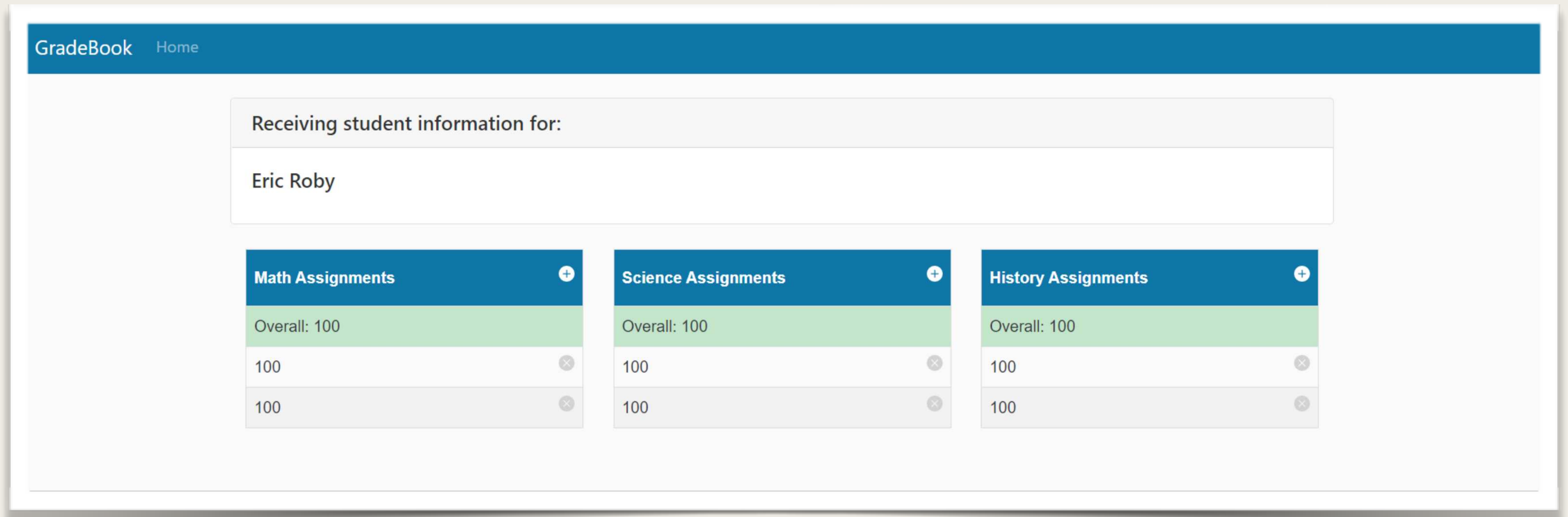
Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100

Add support for tracking grades

To Do: Grades

- Apply TDD to add this new functionality
- Update StudentAndGradeService and DAOs to track grades

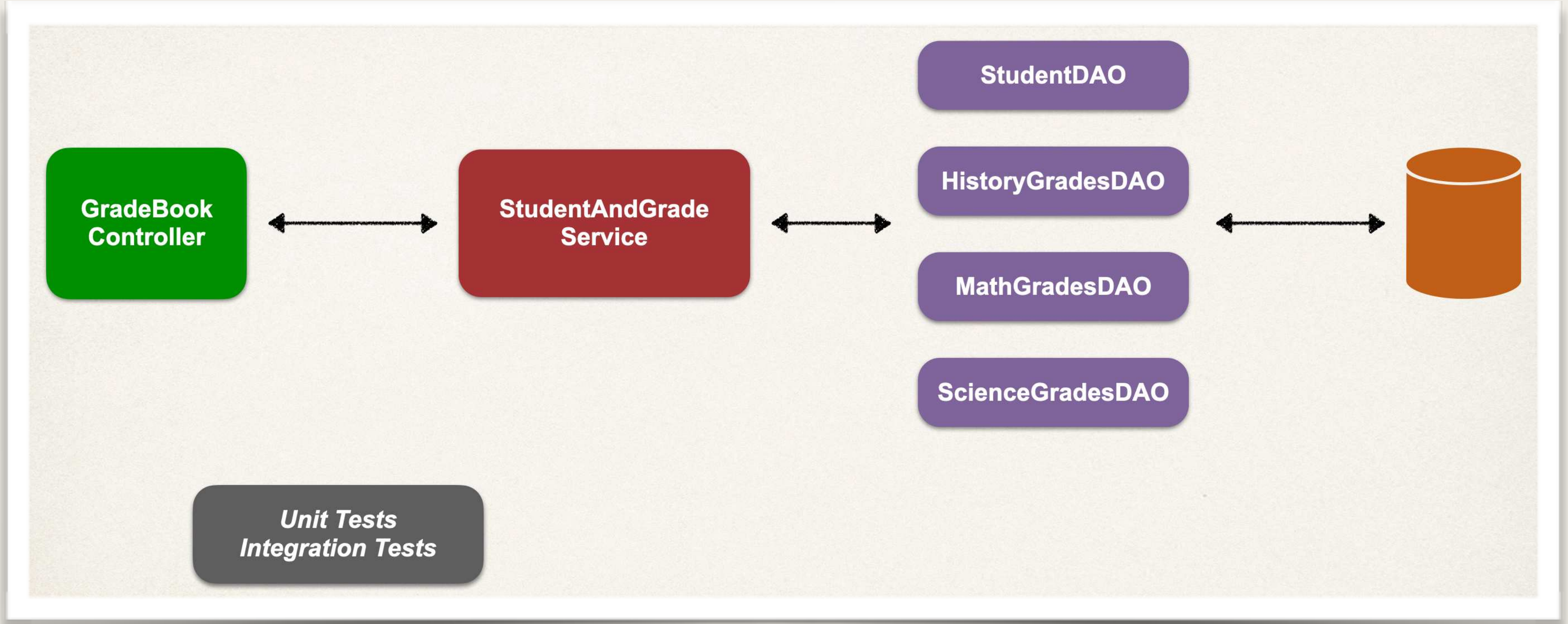


GradeBook Home

Receiving student information for:

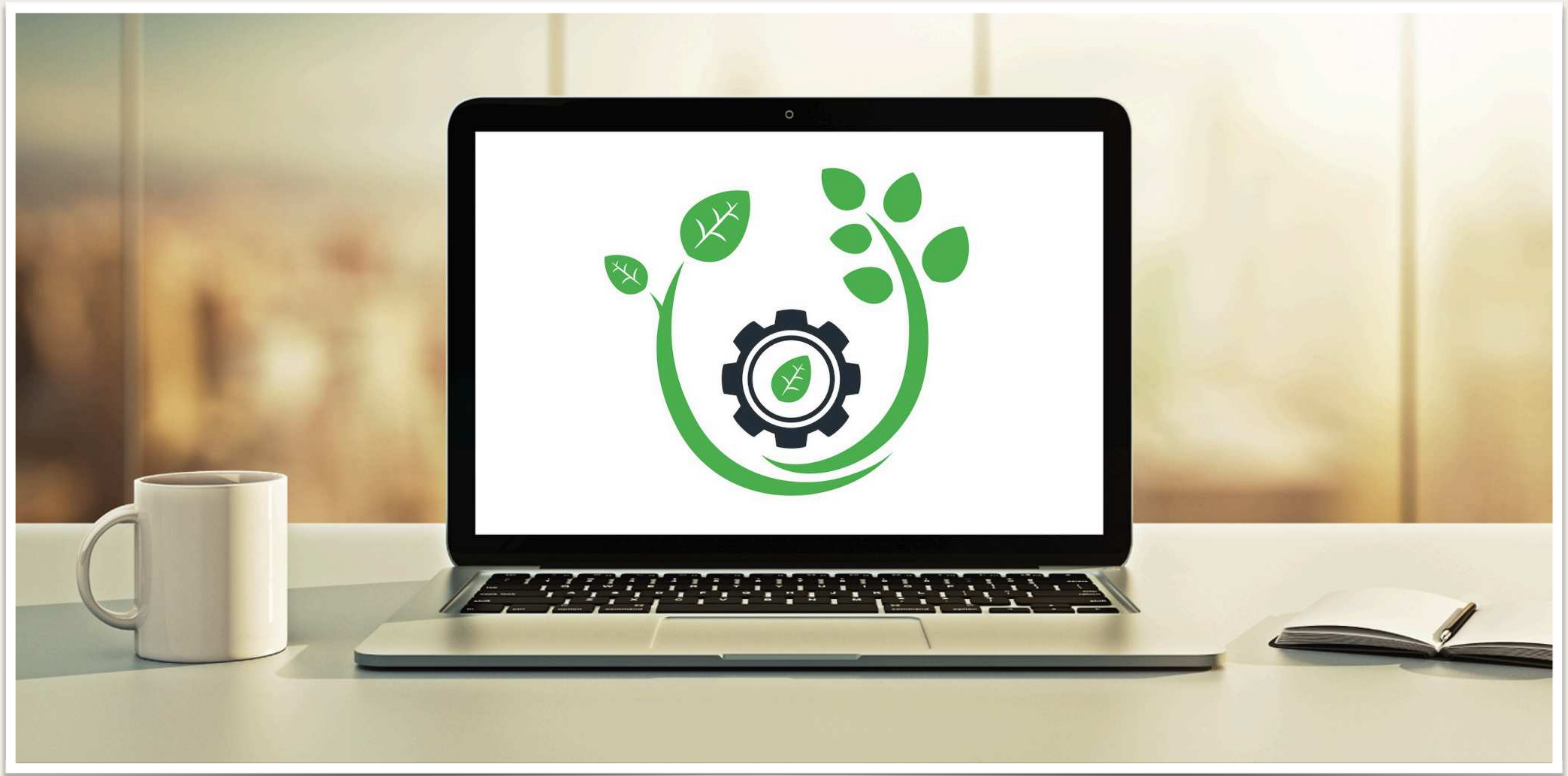
Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100



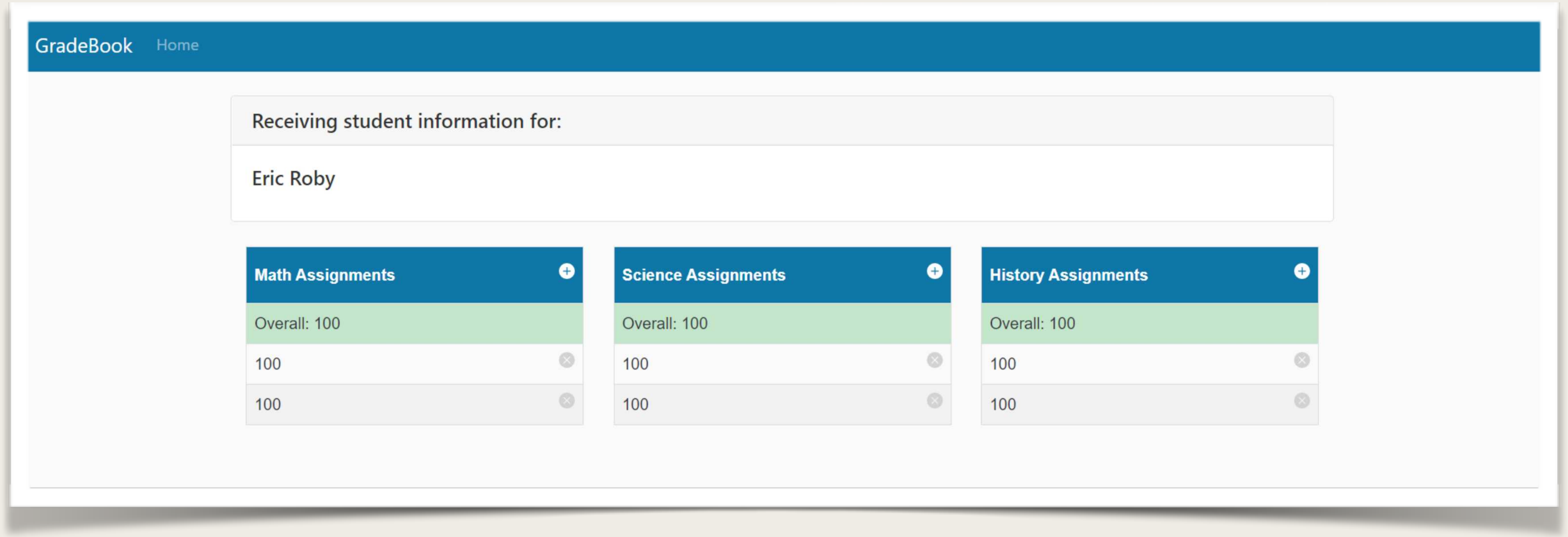
luv2code

Delete Grades



To Do: Delete Grades

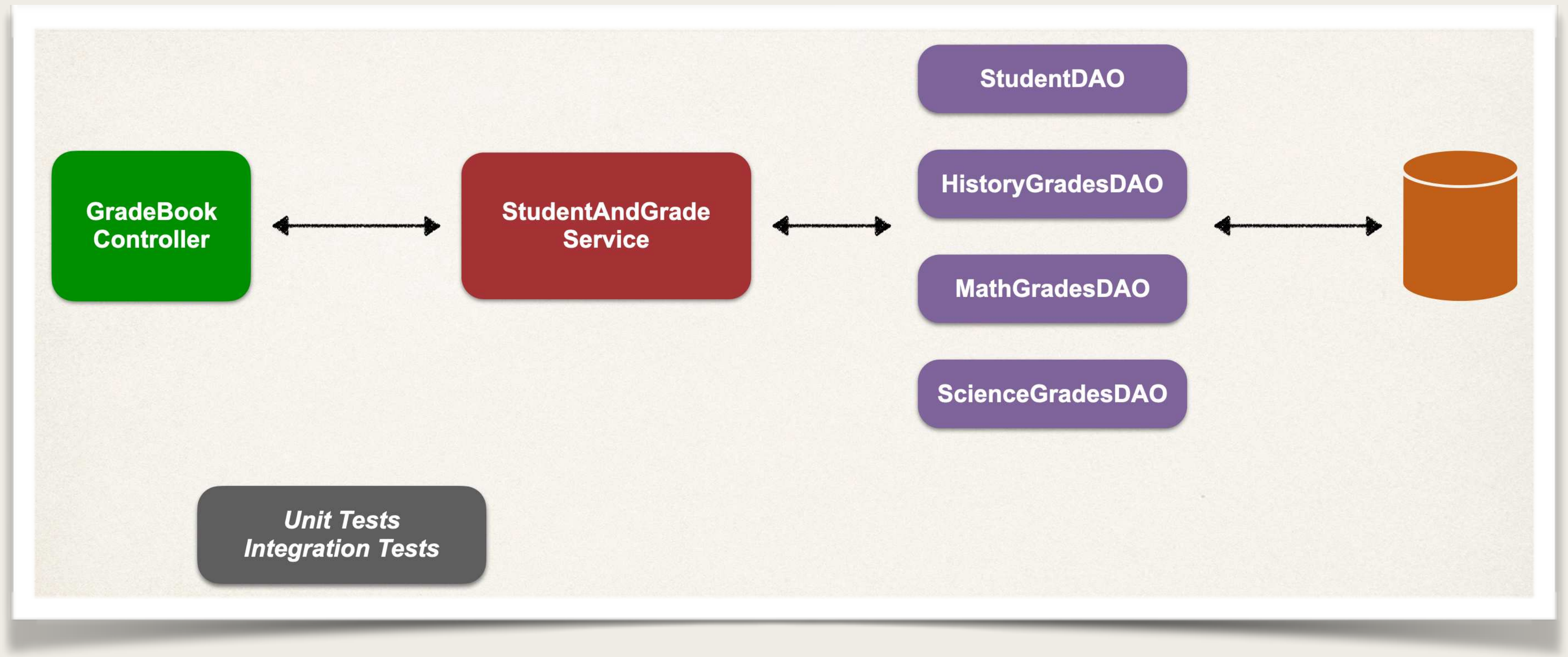
- Currently the app does not delete grades
- Apply TDD to implement this new functionality
- Focus on the backend for now ... we'll come back to UI later


GradeBook Home

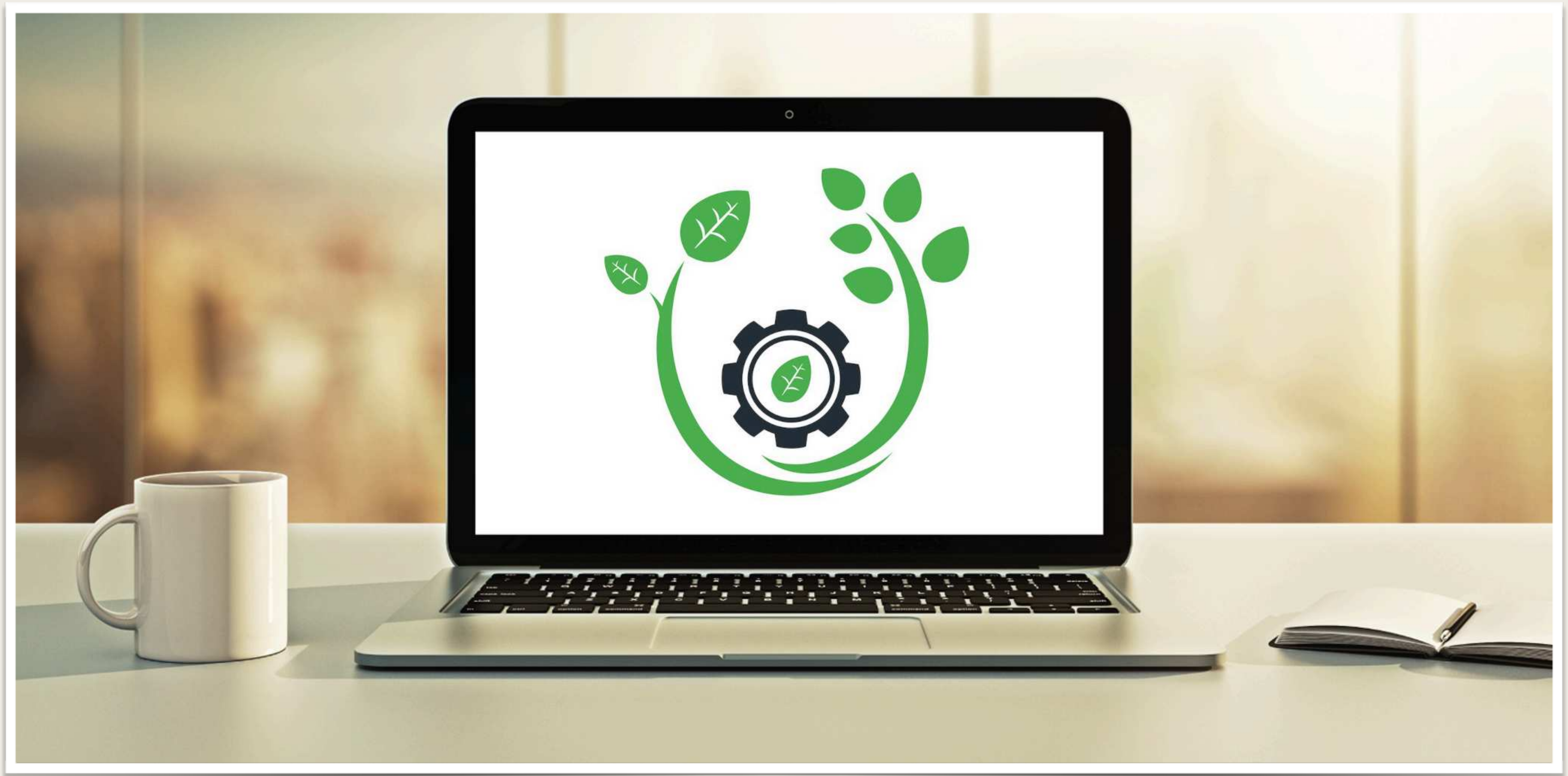
Receiving student information for:

Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100

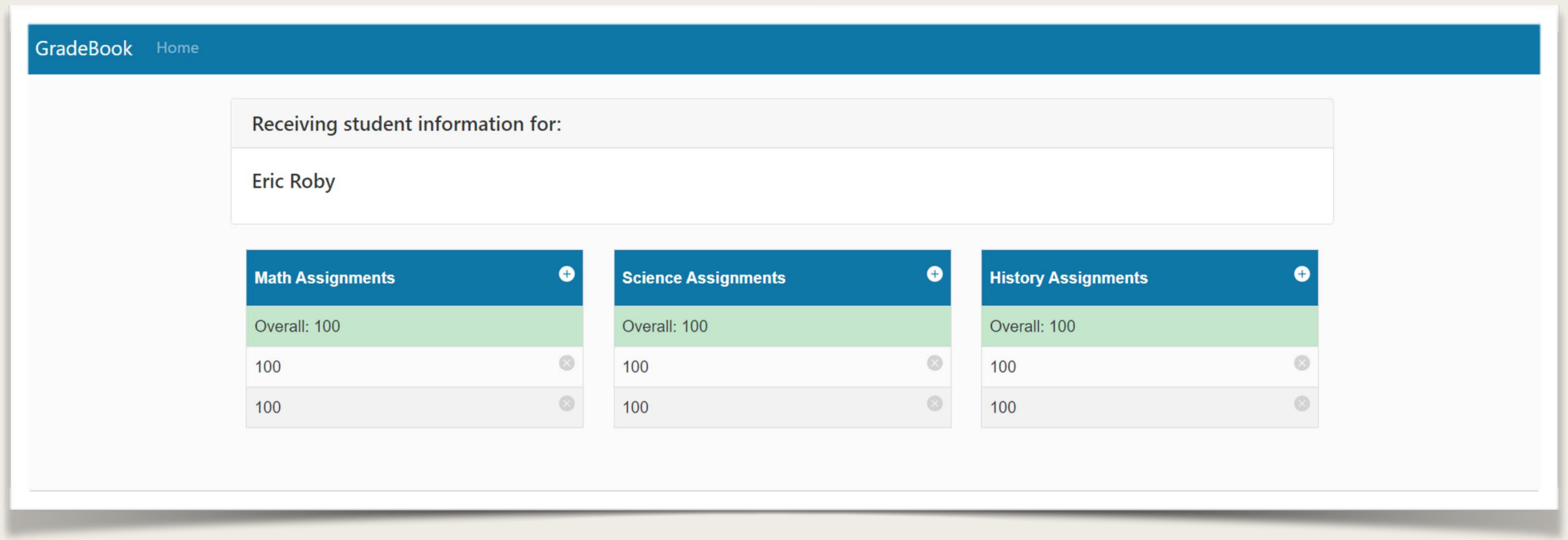


Student Information



To Do: Student Information

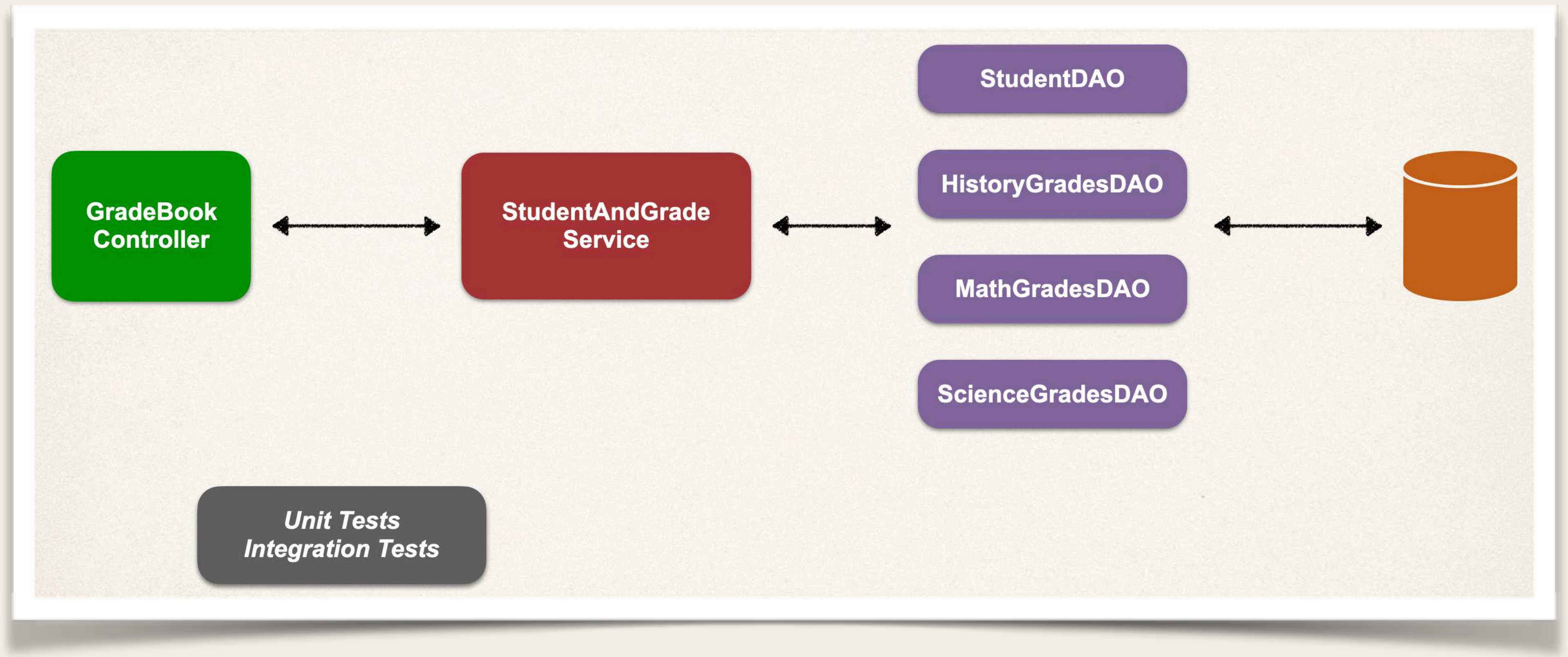
- Currently the app does not have a method to retrieve student information
    - Student name, email, grades etc
- Apply TDD to implement this new functionality



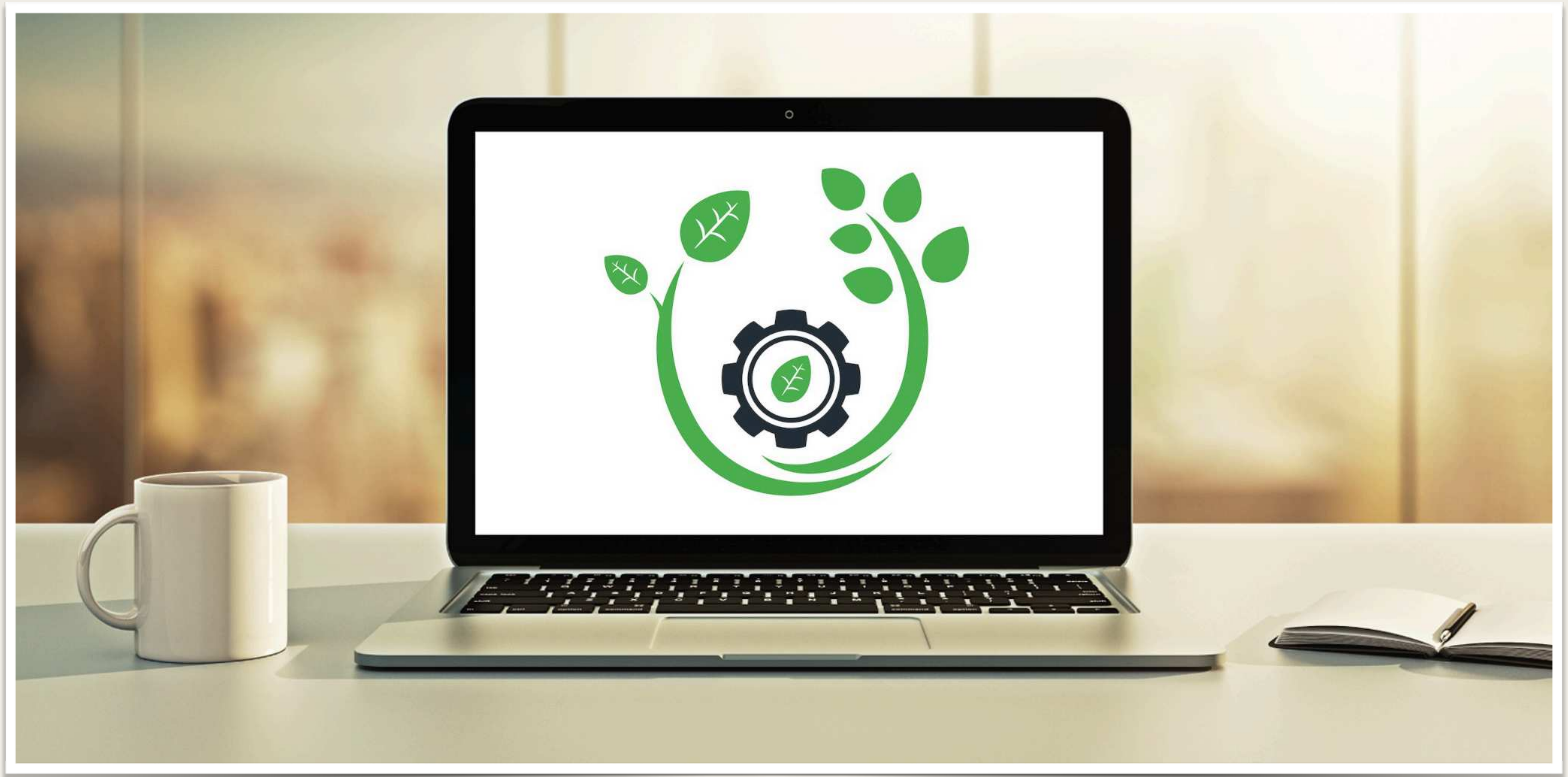
Receiving student information for:

Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100



Set Up SQL Scripts in properties file



SQL for Sample Data

- Currently the SQL for sample data is hard-coded in our tests
- We would like to move the SQL to our properties file

@BeforeEach and @AfterEach

StudentAndGradeServiceTest.java

```java
import org.springframework.jdbc.core.JdbcTemplate;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
...

@TestPropertySource("/application.properties")
@SpringBootTest
public class StudentAndGradeServiceTest {

   @Autowired
   private JdbcTemplate jdbc;

   @BeforeEach
   public void setupDatabase() {
       jdbc.execute("insert into student(id, firstname, lastname, email_address) " +
               "values (1, 'Eric', 'Roby', 'eric.roby@luv2code_school.com')");
       ...
   }


   @AfterEach
   public void setupAfterTransaction() {
       jdbc.execute("DELETE FROM student");
       ...
   }

}
```

This is what we currently have

Insert sample data

Delete sample data

SQL is hard coded

Development Process

Step-By-Step

1. Add SQL to application.properties
2. Inject SQL into test using @Value
3. Refactor @BeforeEach and @AfterEach

Step 1: Add SQL to application.properties

application.properties

```
...

sql.script.create.student=insert into student(id,firstname,lastname,email_address) \
  values (1,'Eric', 'Roby', 'eric.roby@luv2code_school.com')

sql.script.delete.student=DELETE FROM student

...
```

Step 2: Inject SQL into test using @Value

StudentAndGradeServiceTest.java

```java
...

@TestPropertySource("/application.properties")
@SpringBootTest
public class StudentAndGradeServiceTest {

   @Autowired
   private JdbcTemplate jdbc;

   @Value("${sql.script.create.student}")
   private String sqlAddStudent;

   @Value("${sql.script.delete.student}")
   private String sqlDeleteStudent;

   ...
}
```

application.properties

```
...

sql.script.create.student=insert into student(id,firstname,lastname,email_address) \
  values (1,'Eric', 'Roby', 'eric.roby@luv2code_school.com')

sql.script.delete.student=DELETE FROM student

...
```

Step 3: Refactor @BeforeEach and @AfterEach

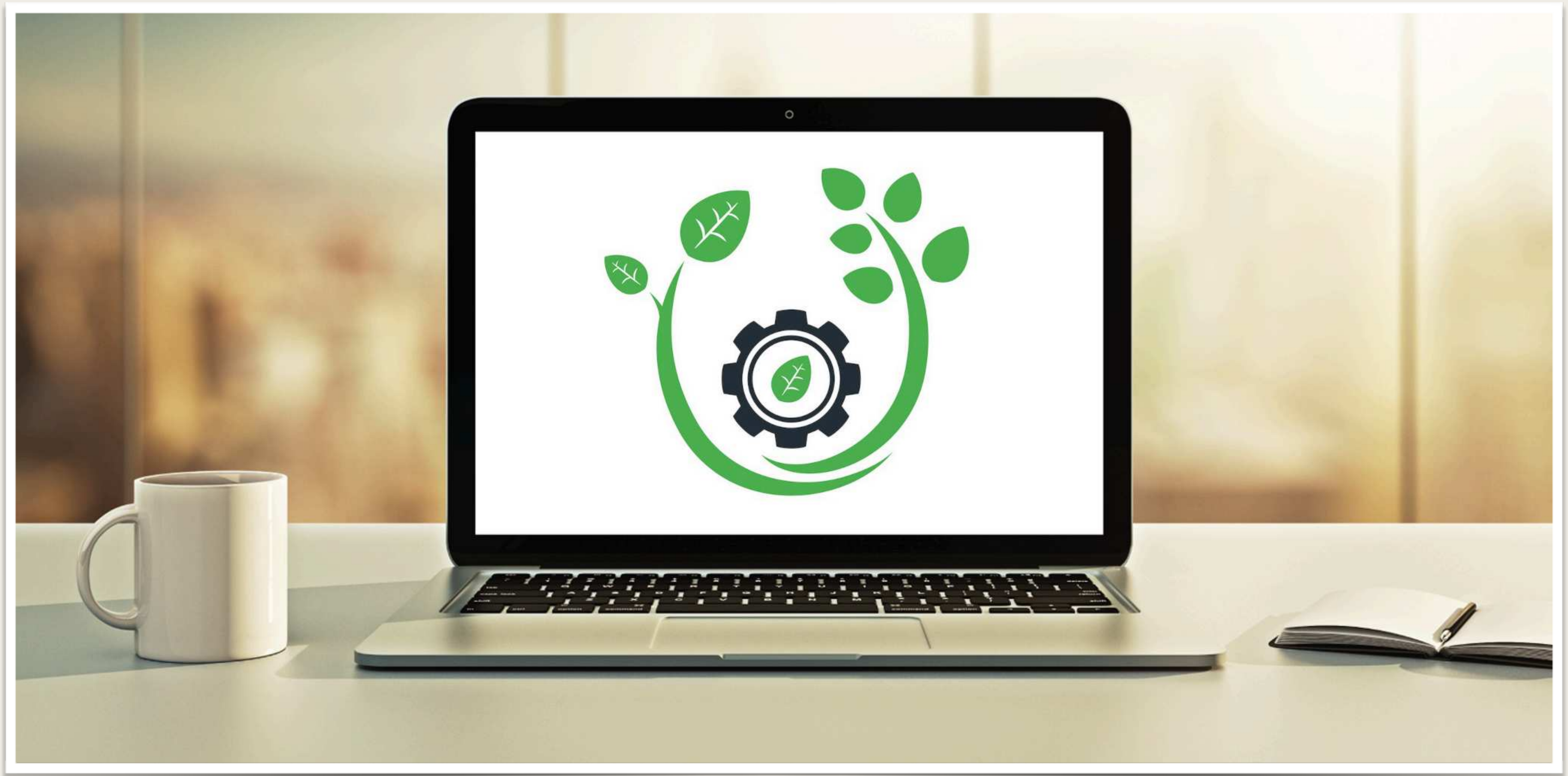
StudentAndGradeServiceTest.java

```java
...

@TestPropertySource("/application.properties")
@SpringBootTest
public class StudentAndGradeServiceTest {

    @Autowired
    private JdbcTemplate jdbc;

    @Value("${sql.script.create.student}")
    private String sqlAddStudent;

    @Value("${sql.script.delete.student}")
    private String sqlDeleteStudent;

    @BeforeEach
    public void setupDatabase() {
        jdbc.execute(sqlAddStudent);
    }


    @AfterEach
    public void setupAfterTransaction() {
        jdbc.execute(sqlDeleteStudent);
    }

}
```

Refactored code

Refactored code

MVC Tests for Student Information

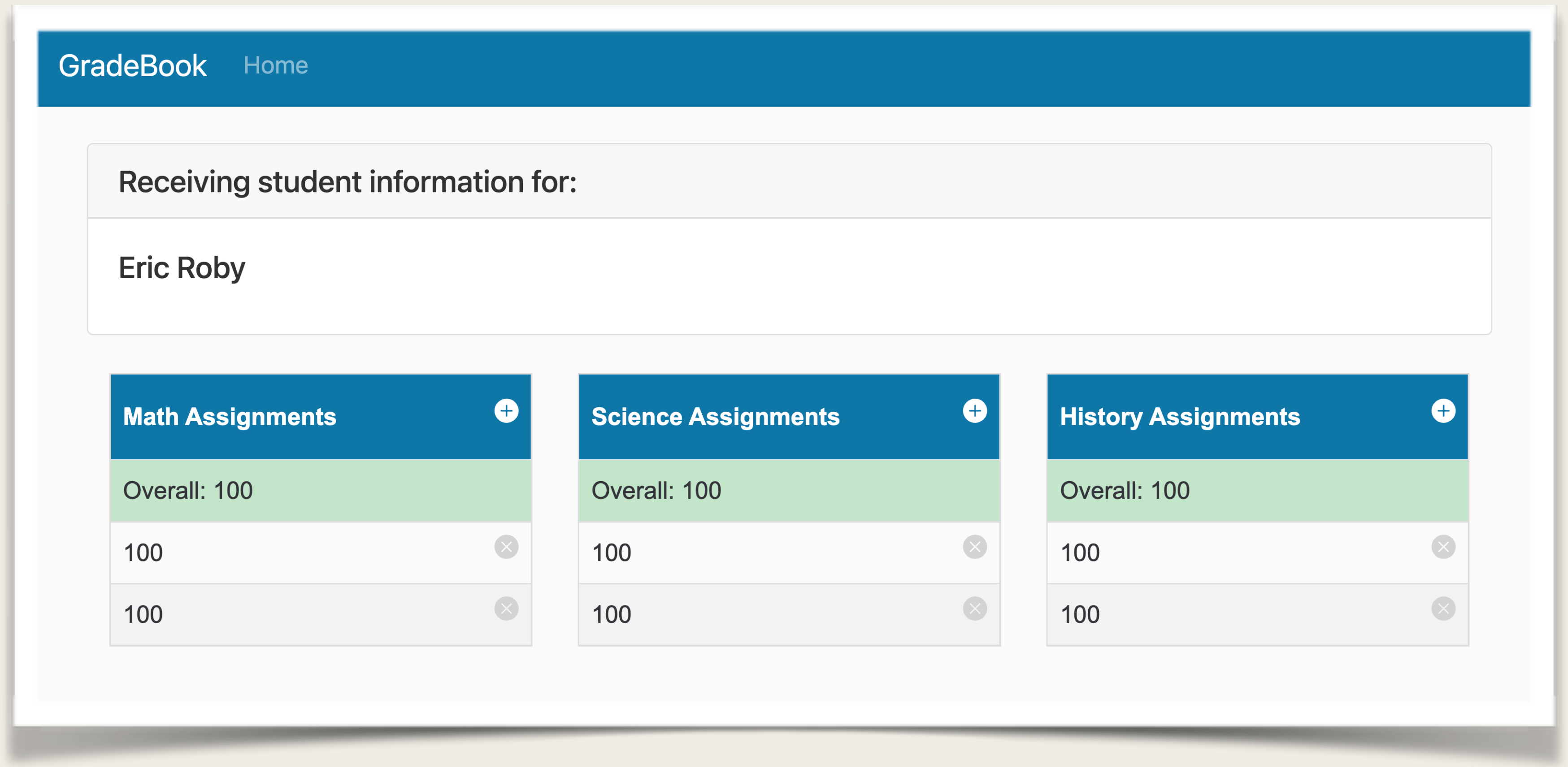


MVC Tests for Student Information

- If we have valid student id, return view name: `studentInformation`
- If we have invalid student id, return view name: `error`

Thymeleaf templates

File: src/main/resources/templates/studentInformation.html



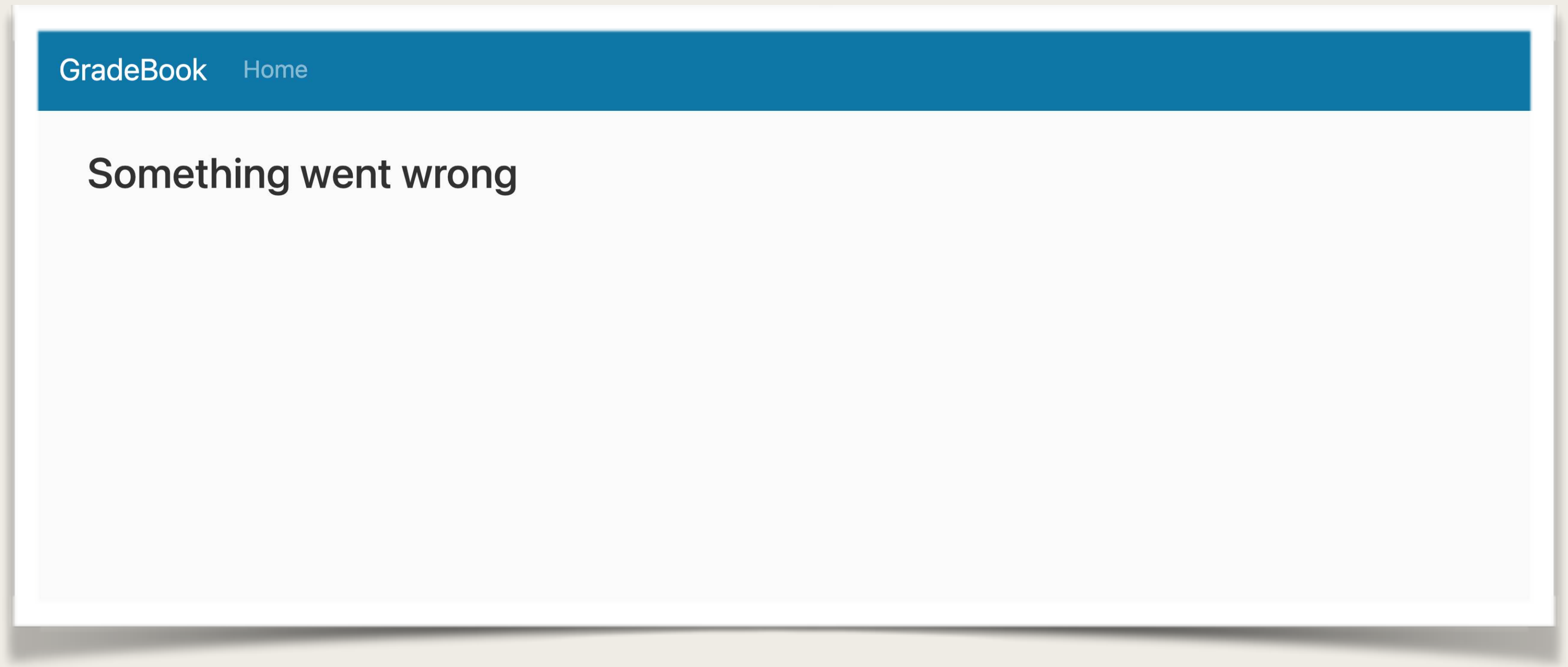
GradeBook Home

Receiving student information for:

Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100

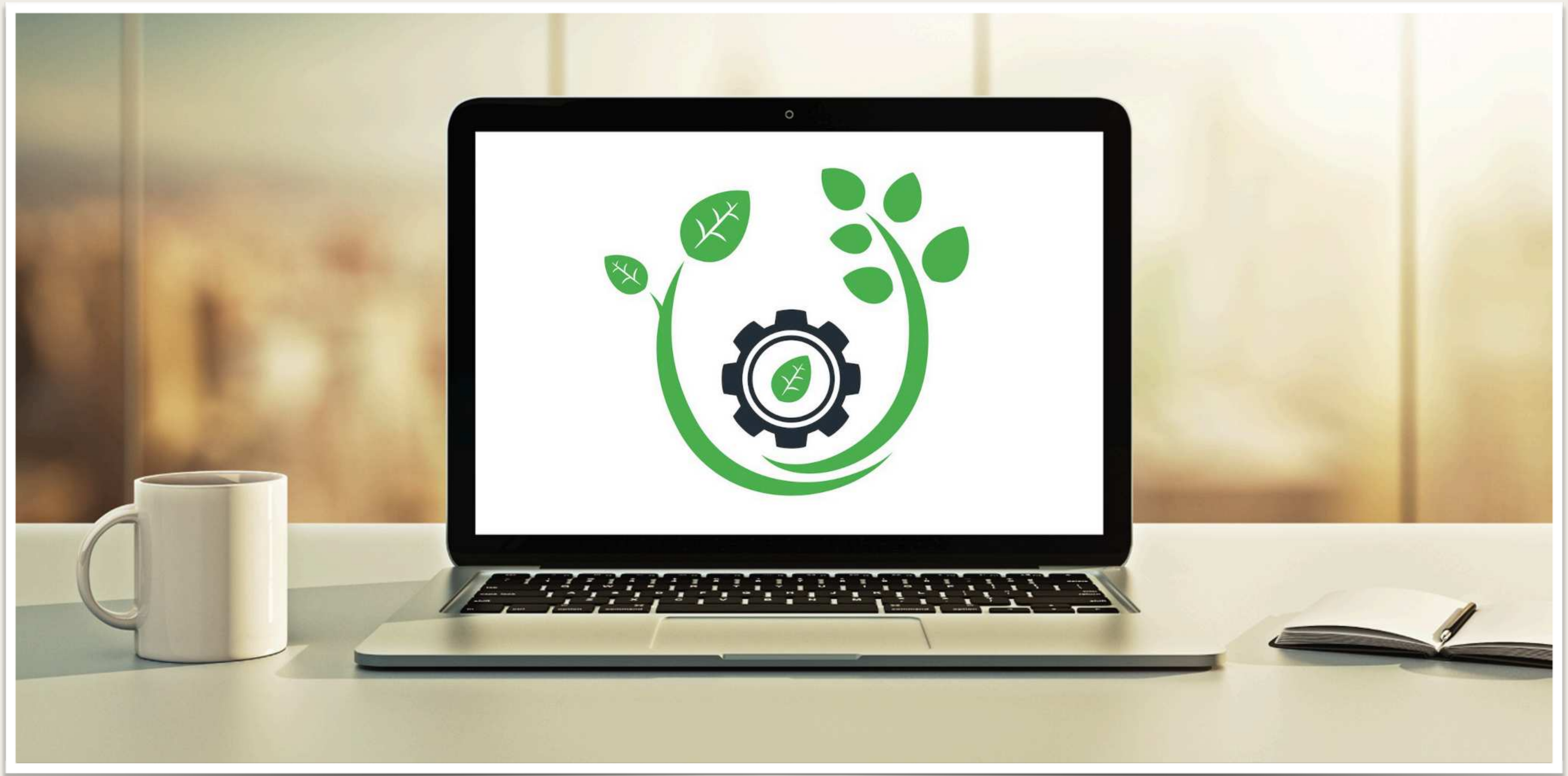
File: src/main/resources/templates/error.html



GradeBook Home

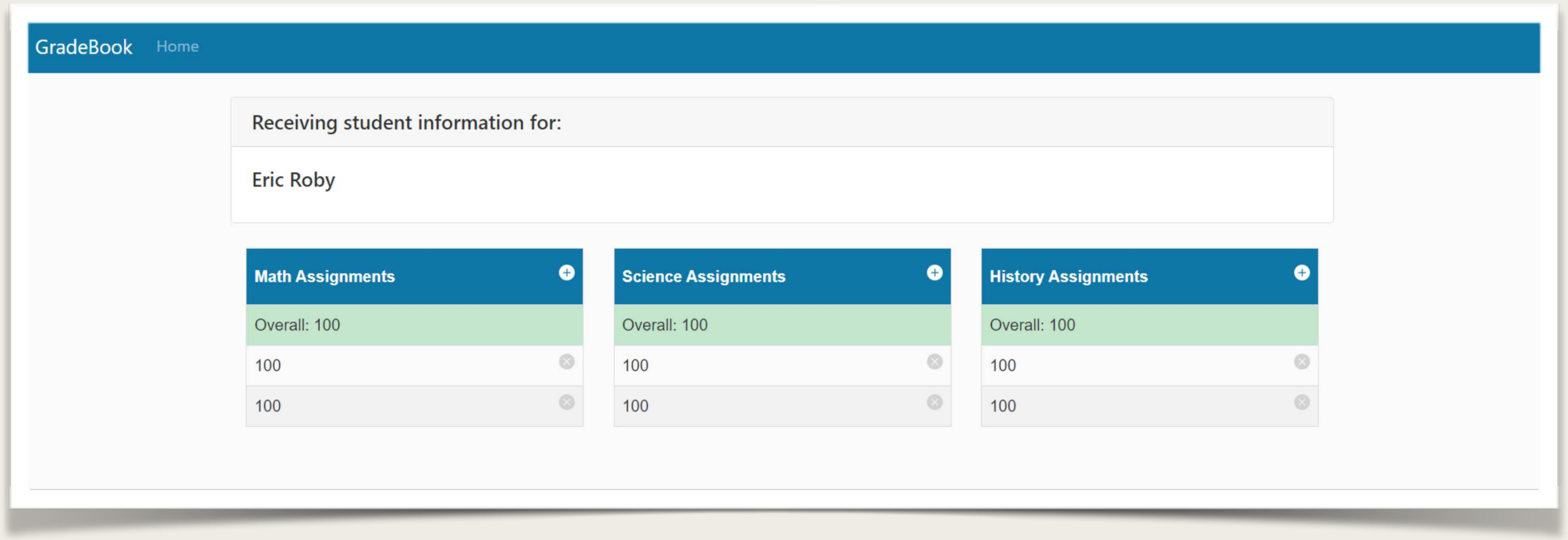
Something went wrong

Create Grades with MVC Controller



To Do: Create Grades

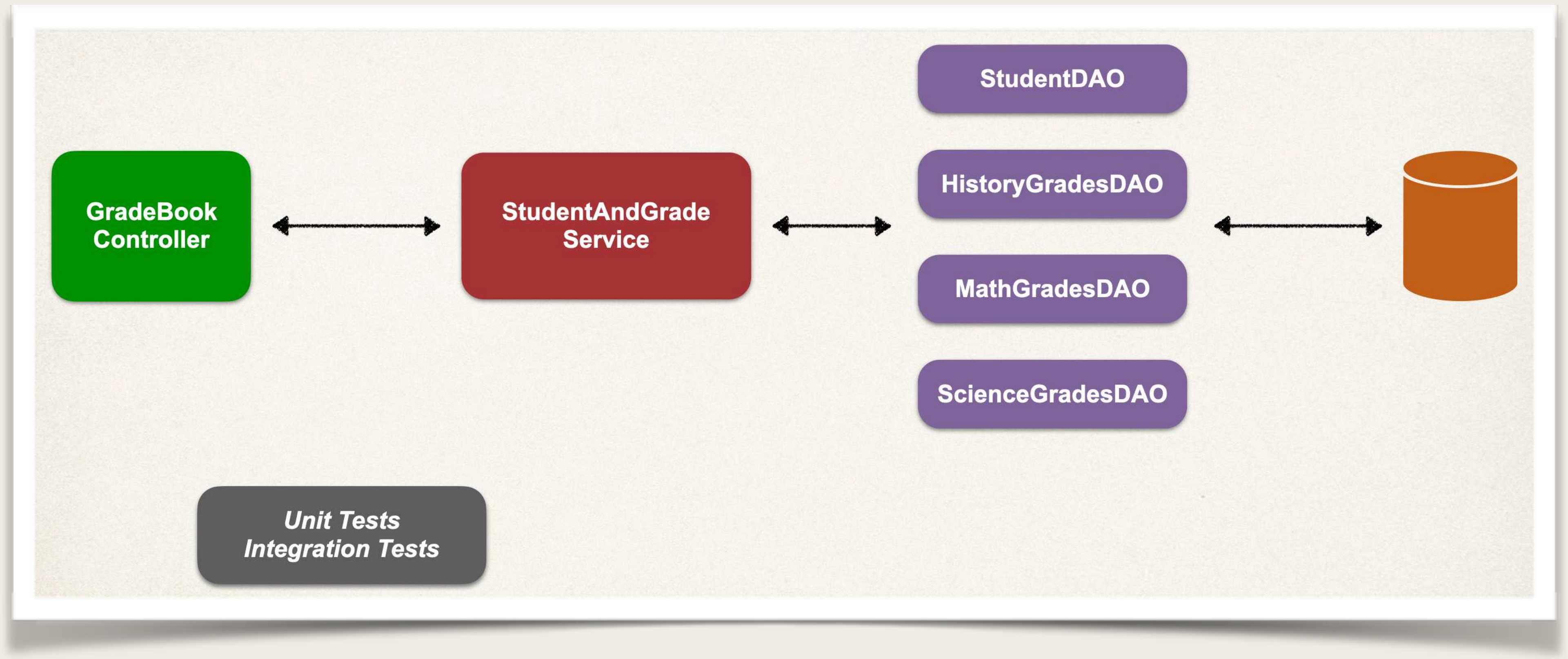
- Currently the app does not support creating new grades via MVC controller
- Apply TDD to add support for this functionality


GradeBook Home

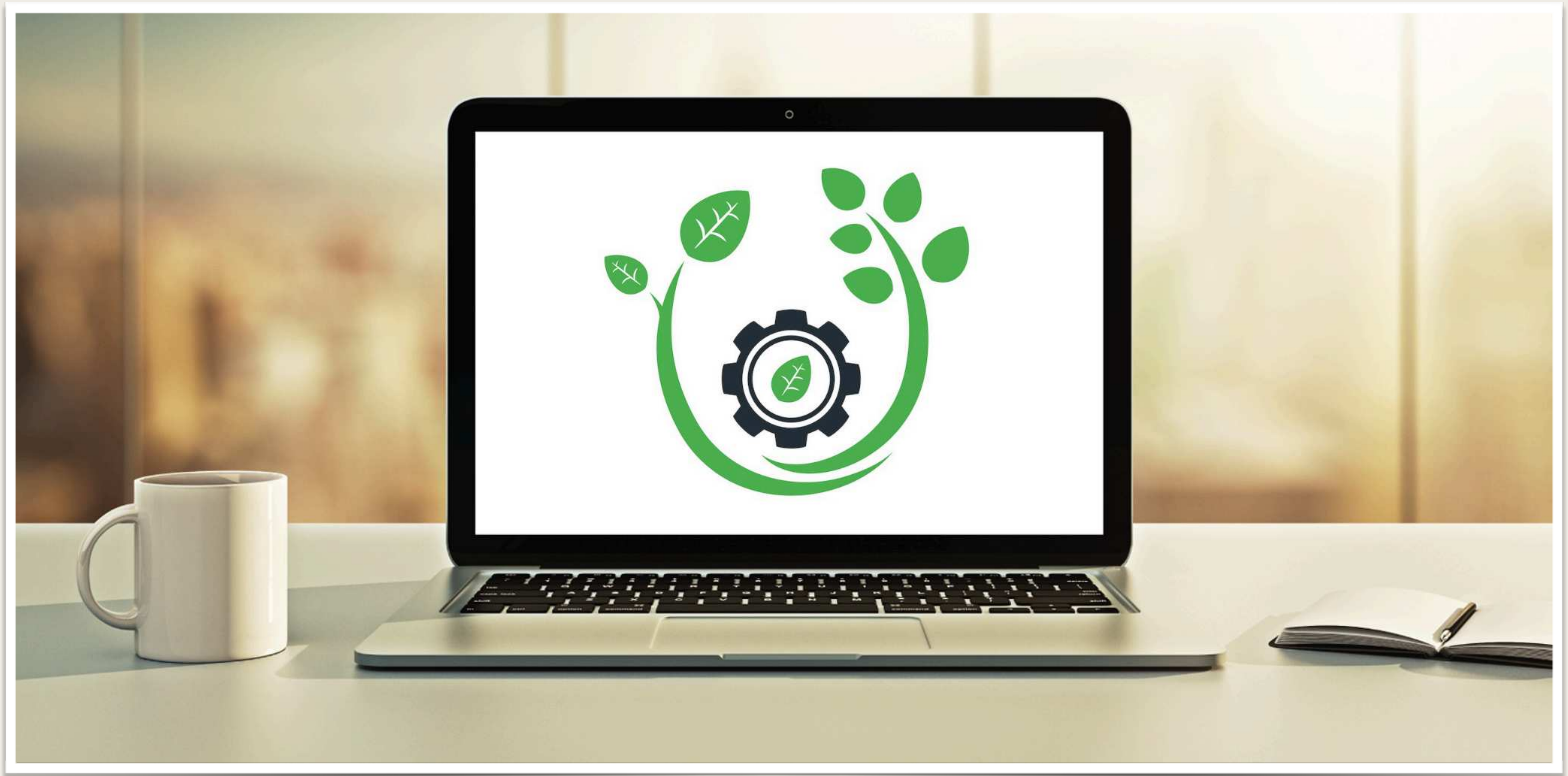
Receiving student information for:

Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100

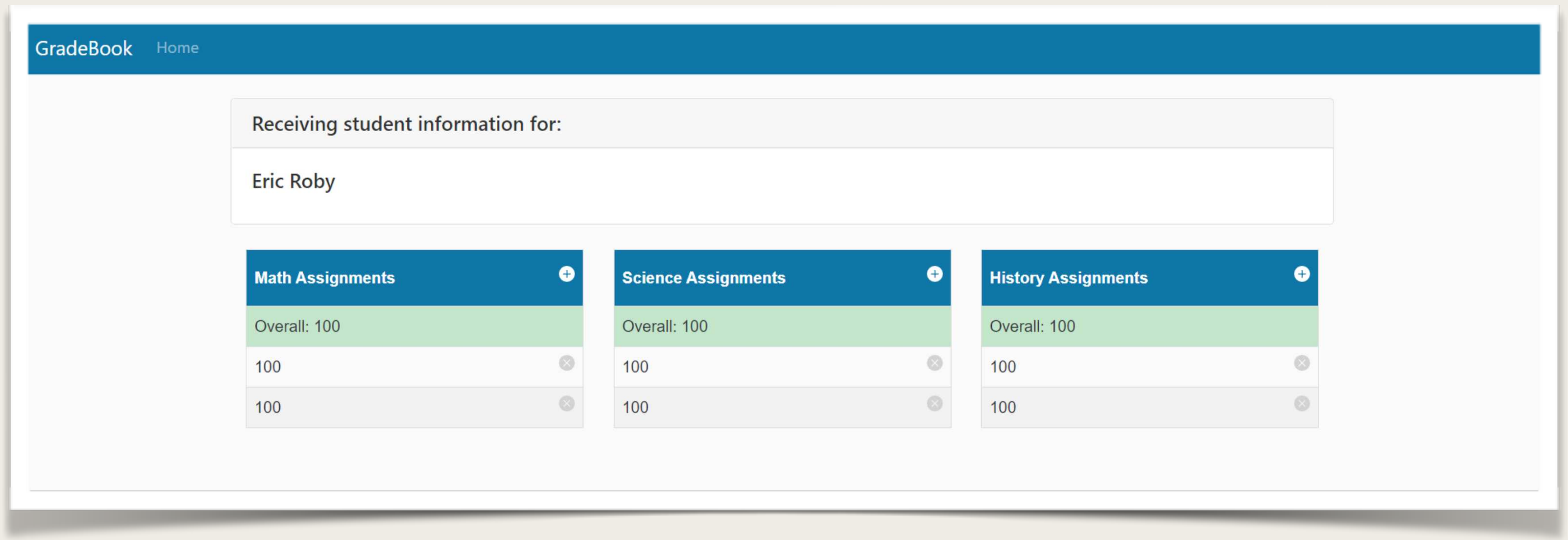


Delete Grades with MVC Controller



To Do: Delete Grades

- Currently the app does not support deleting grades via MVC controller
- Apply TDD to add support for this functionality

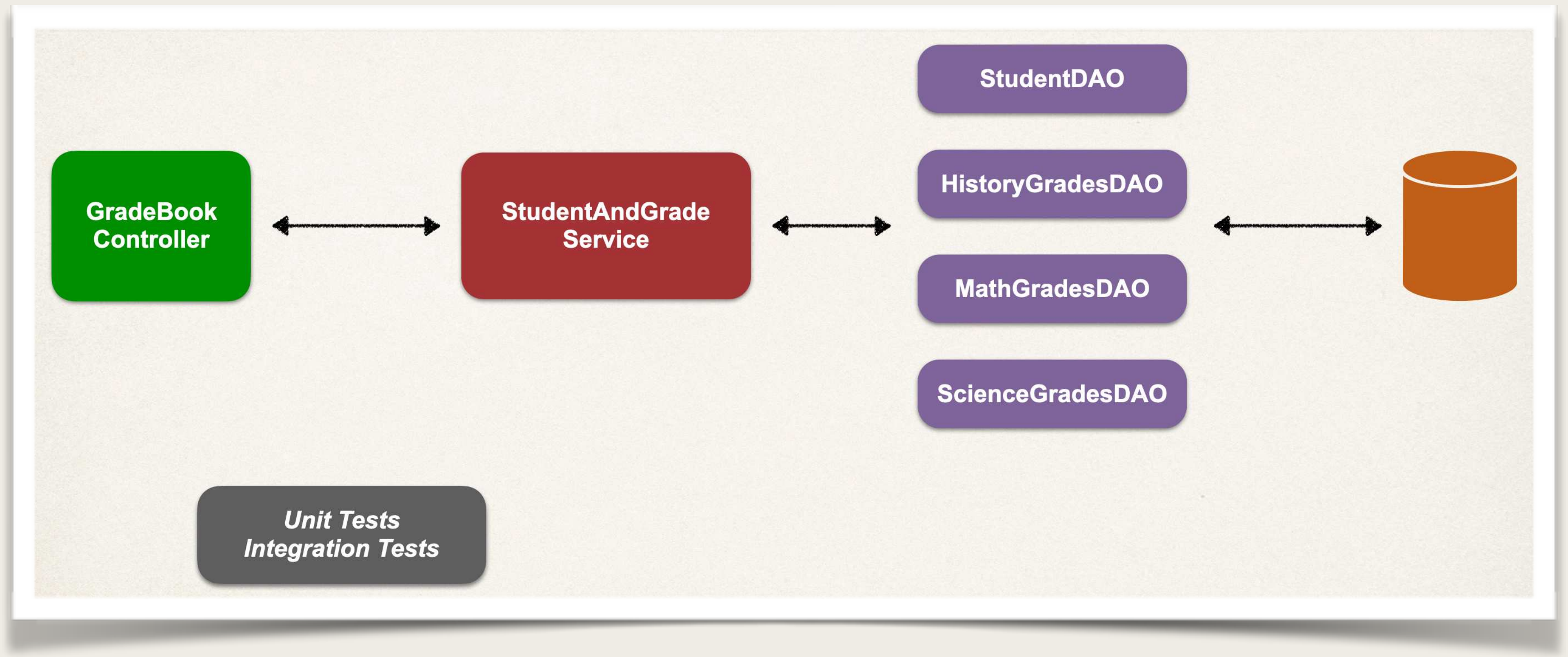


GradeBook Home

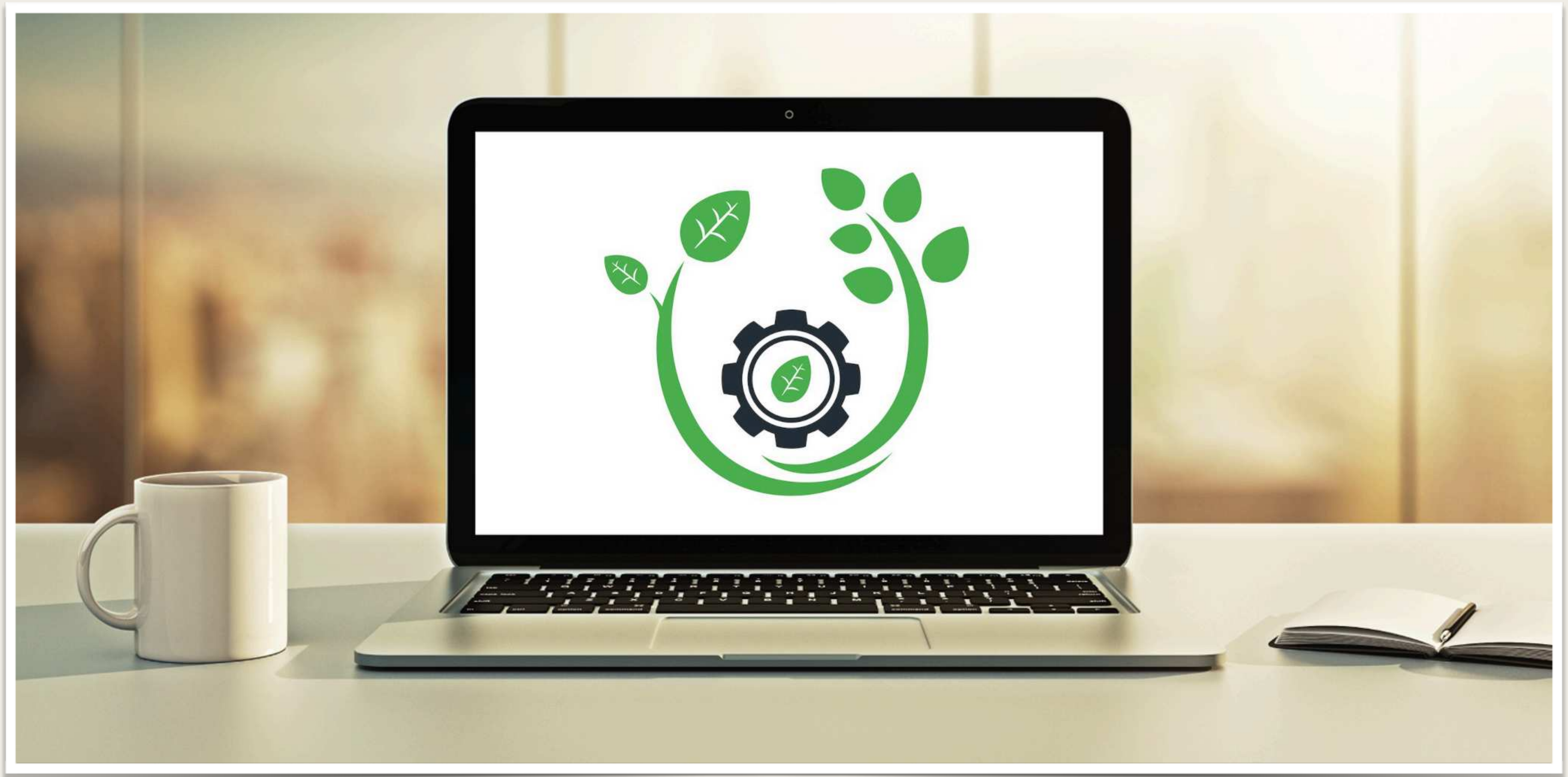
Receiving student information for:

Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100



Update Thymeleaf Template Student Information

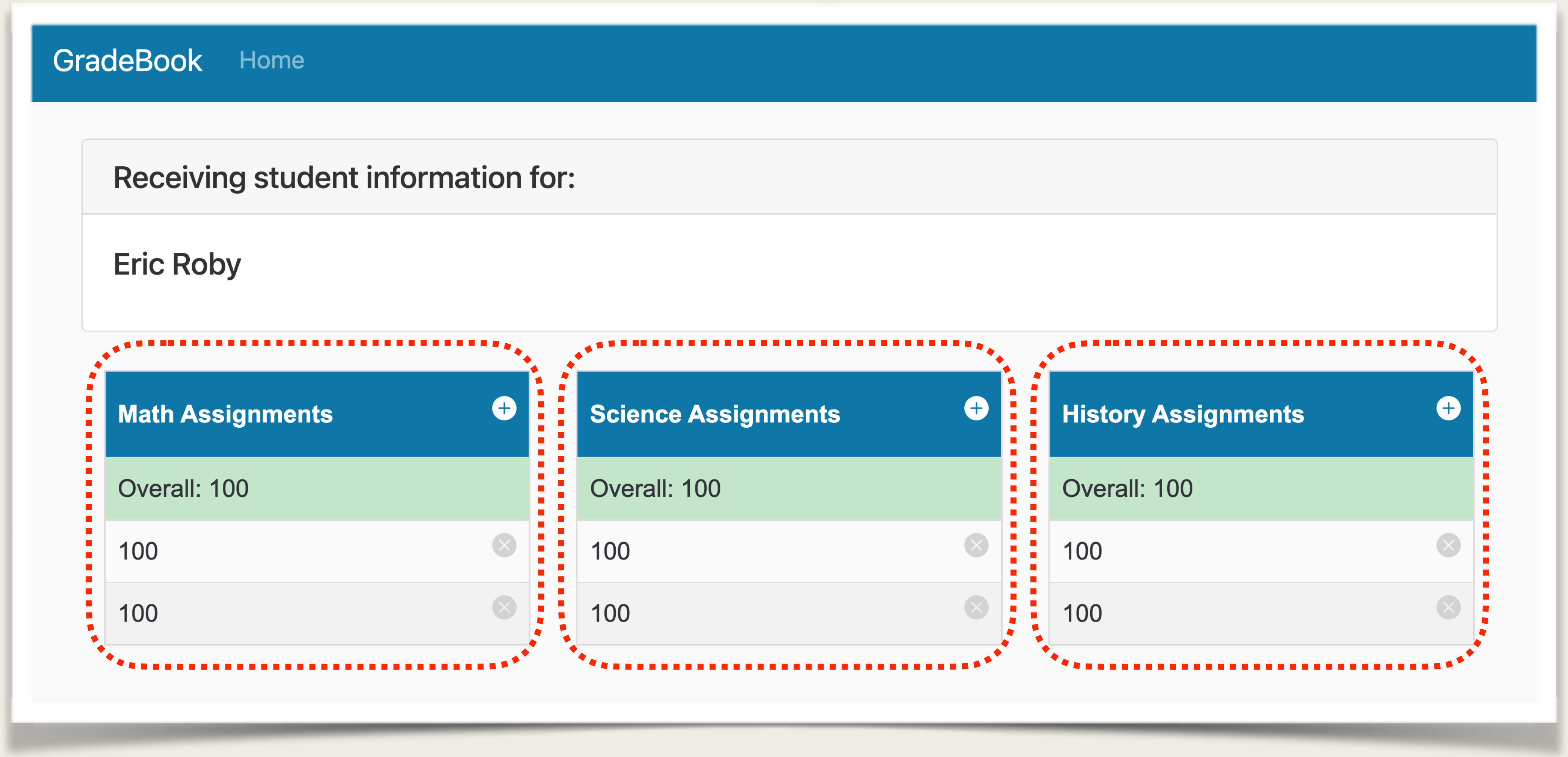


To Do

- Currently the UI for Student Information has hard-coded data ✗
- Update the Thymeleaf template for Student Information to use dynamic data ✓

Thymeleaf template

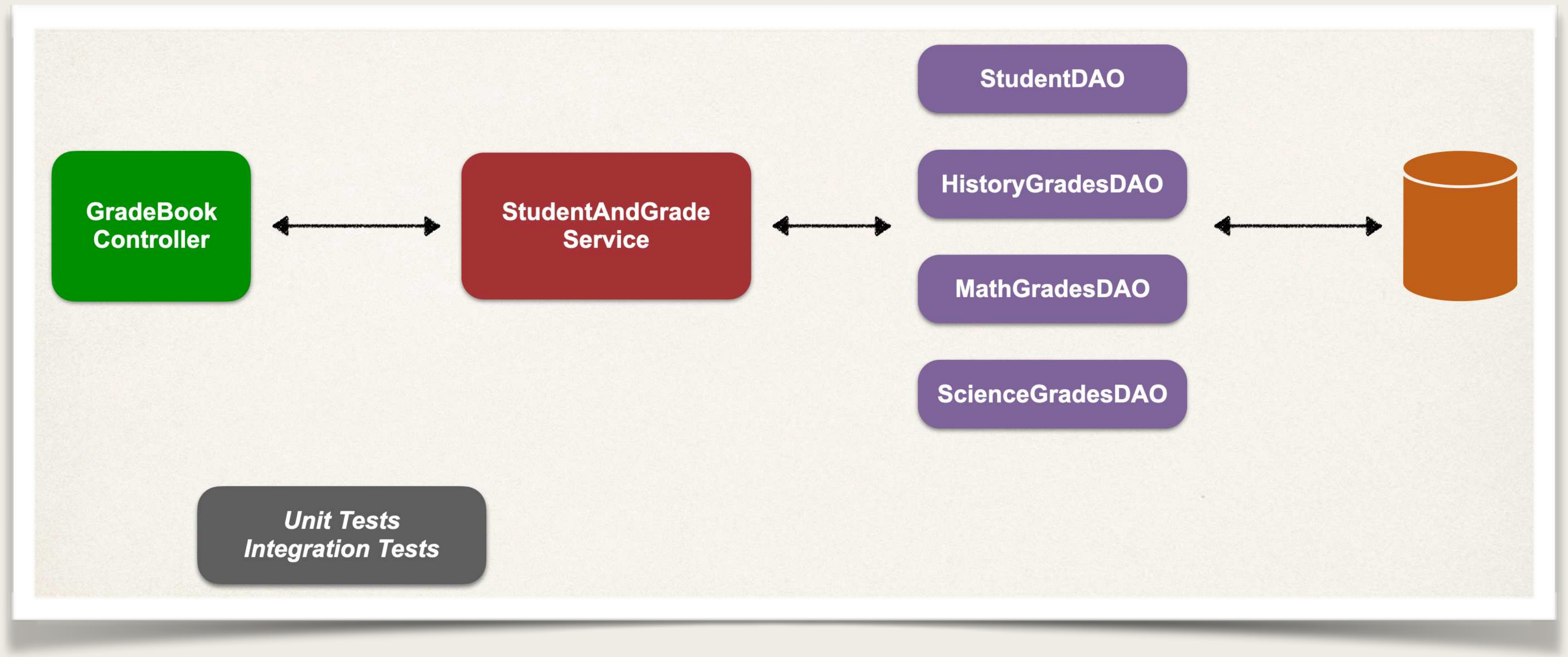
File: src/main/resources/templates/studentInformation.html



Receiving student information for:

Eric Roby

Math Assignments	Science Assignments	History Assignments
Overall: 100	Overall: 100	Overall: 100
100	100	100
100	100	100


GradeBook Controller
StudentAndGrade Service
StudentDAO
HistoryGradesDAO
MathGradesDAO
ScienceGradesDAO
Unit Tests
Integration Tests