

# 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

```
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(firstname, lastname, email_address) " +
            "values ('Eric', 'Roby', 'eric.roby@luv2code_school.com')");
        ...
    }

    @AfterEach
    public void setupAfterTransaction() {
        jdbc.execute("DELETE FROM student");
        jdbc.execute("ALTER TABLE student ALTER COLUMN ID RESTART WITH 1");
        ...
    }
}
```

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(firstname,lastname,email_address) \  
    values ('Eric', 'Roby', 'eric.robby@luv2code_school.com')
```

```
sql.script.delete.student=DELETE FROM student; ALTER TABLE student ALTER COLUMN ID RESTART WITH 1
```

...



# Step 2: Inject SQL into test using @Value

StudentAndGradeServiceTest.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(firstname,lastname,email_address) \
    values ('Eric', 'Roby', 'eric.robby@luv2code_school.com')

sql.script.delete.student=DELETE FROM student; ALTER TABLE student ALTER COLUMN ID RESTART WITH 1

...
```



# Step 3: Refactor @BeforeEach and @AfterEach

StudentAndGradeServiceTest.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



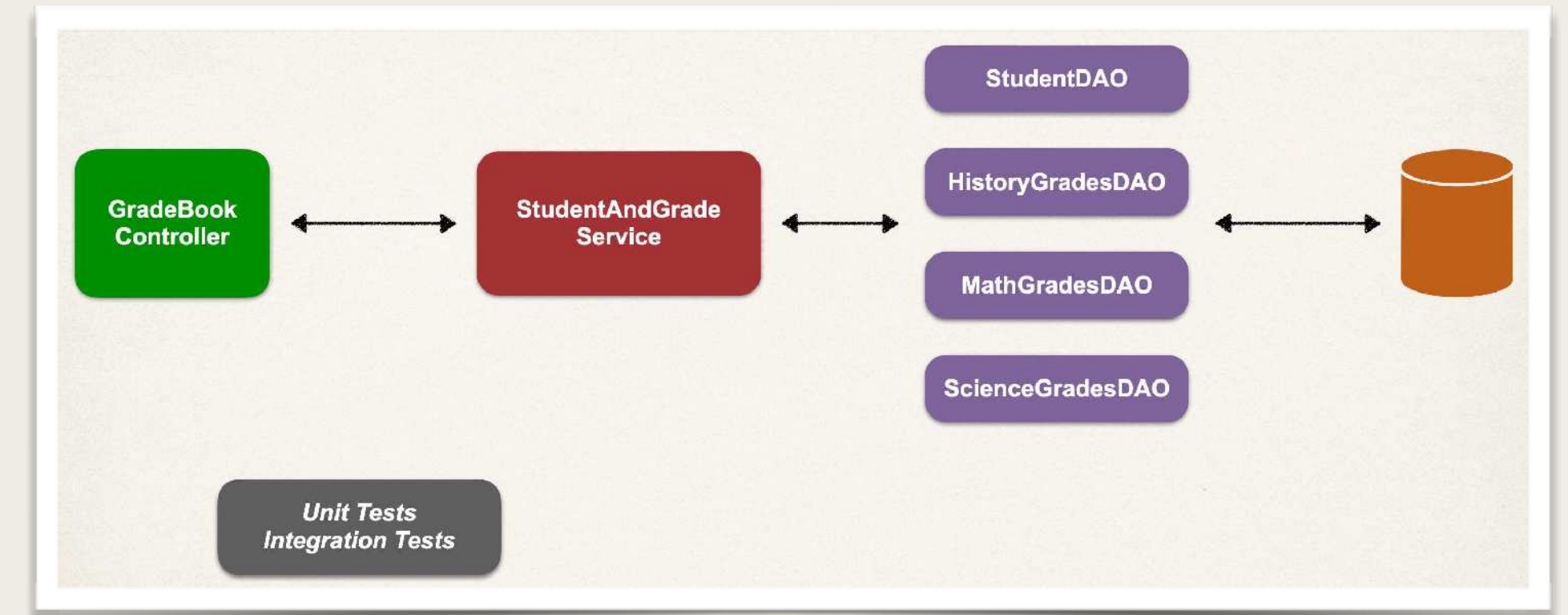
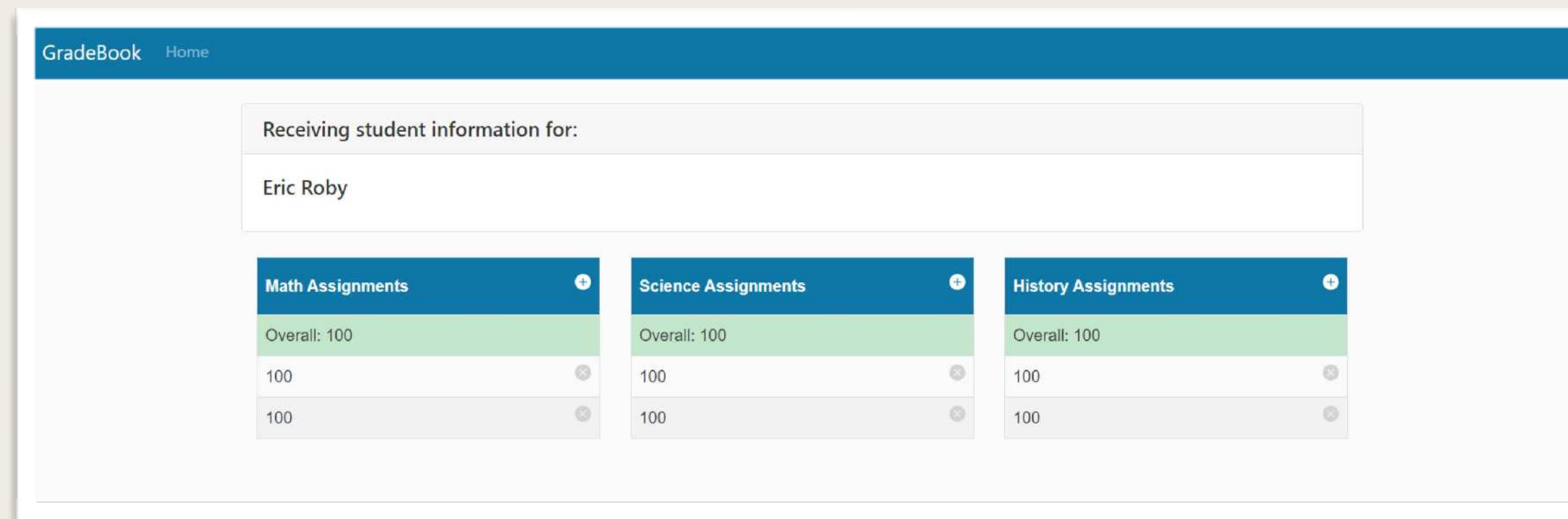
# Create Grades with MVC Controller





# To Do: Create Grades

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





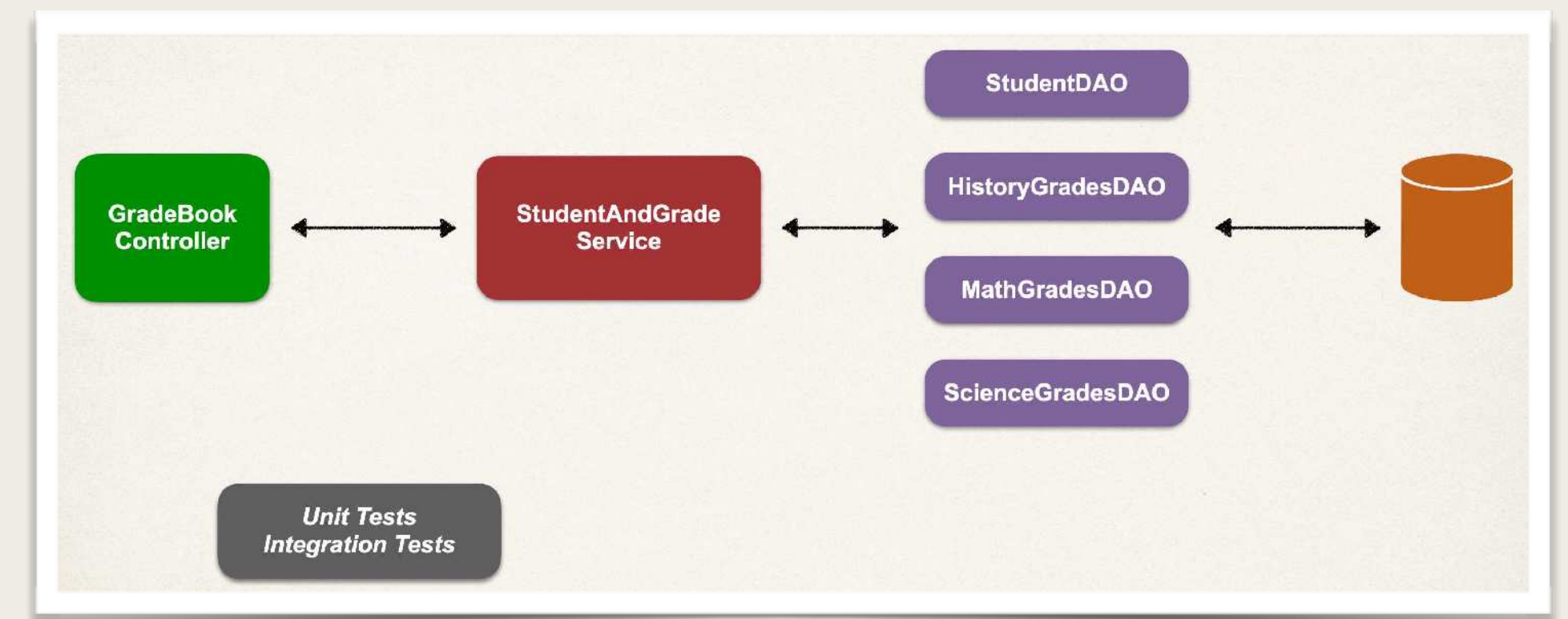
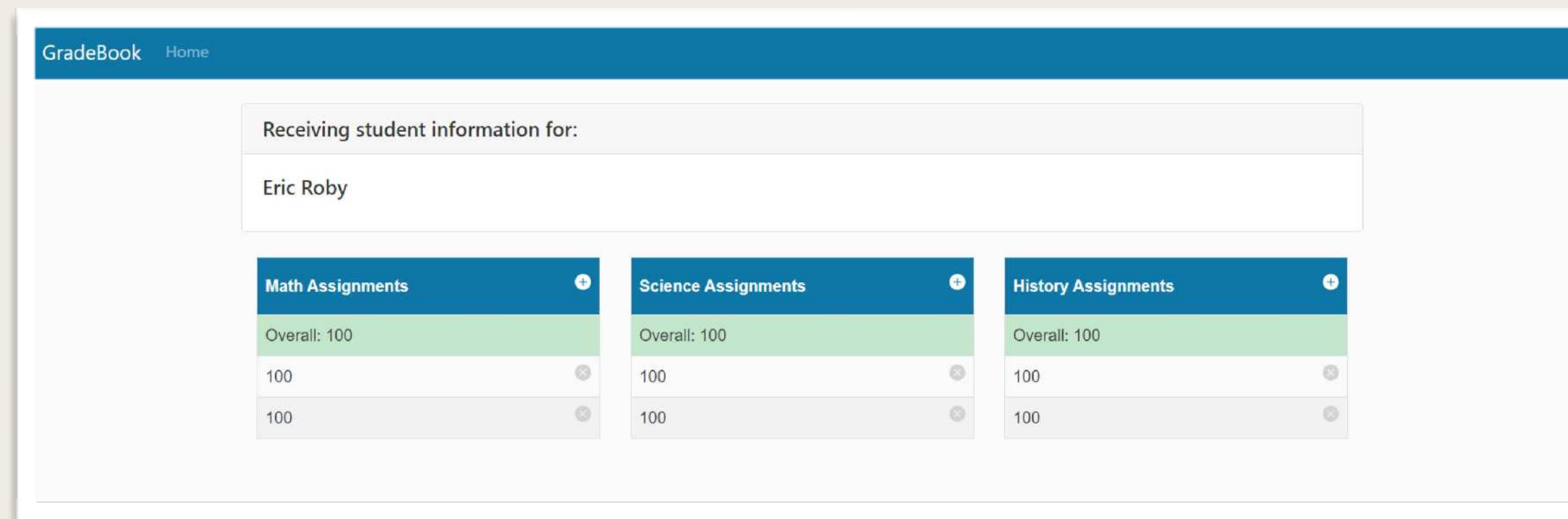
# Delete Grades with MVC Controller





# To Do: Delete Grades

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





# 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`

The screenshot shows a web application interface for a gradebook. At the top, there's a blue header with "GradeBook" and "Home" links. Below it, a section titled "Receiving student information for:" contains the name "Eric Roby". Underneath, there are three tables: "Math Assignments", "Science Assignments", and "History Assignments". Each table has a blue header with a "+" icon, a green row for "Overall: 100", and two rows of "100" scores. The tables are enclosed in a red dashed border, indicating they are the focus of the update.

