Plagiarism Detector

Phase C: Final Implementation

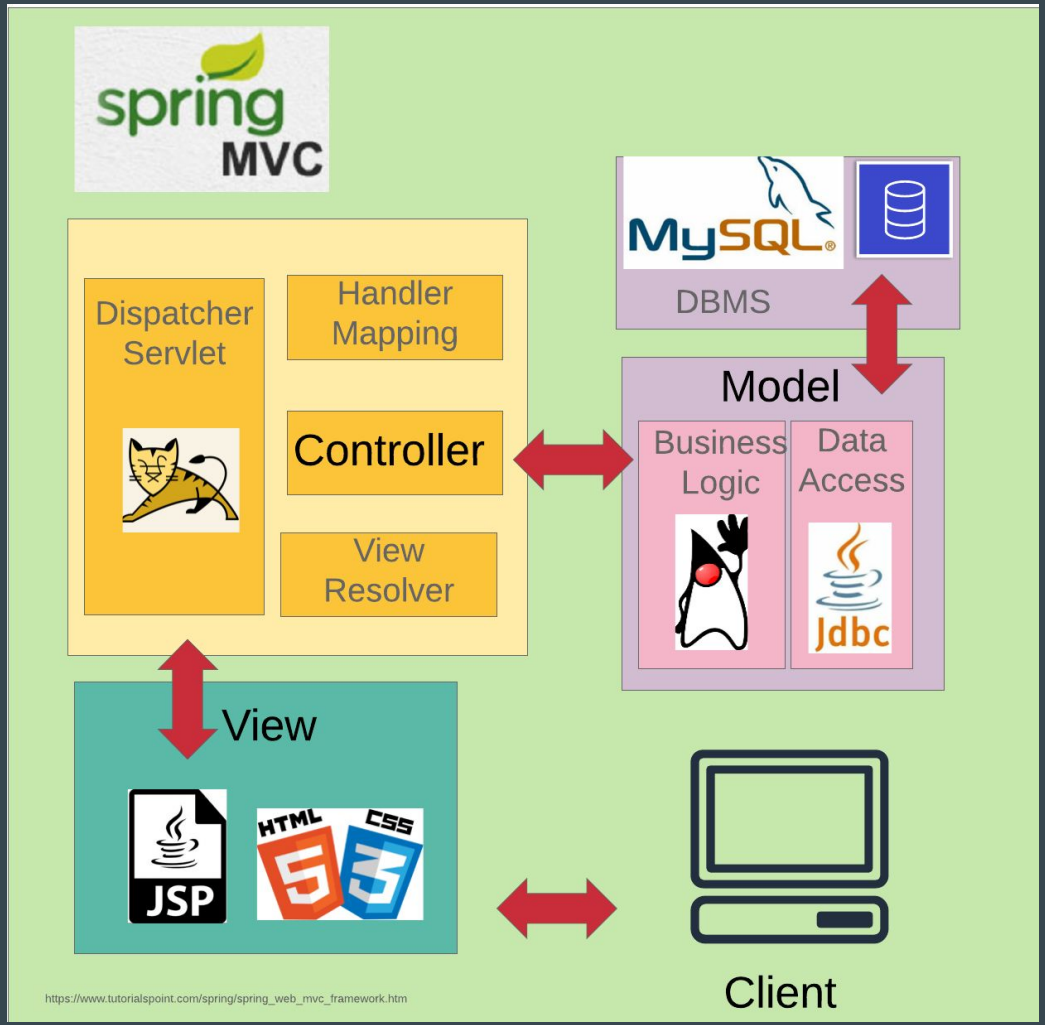
Group 2

Shagun Bhardwaj
Kenji Fujita
Mason Leon
Prakash Tarun Kumar

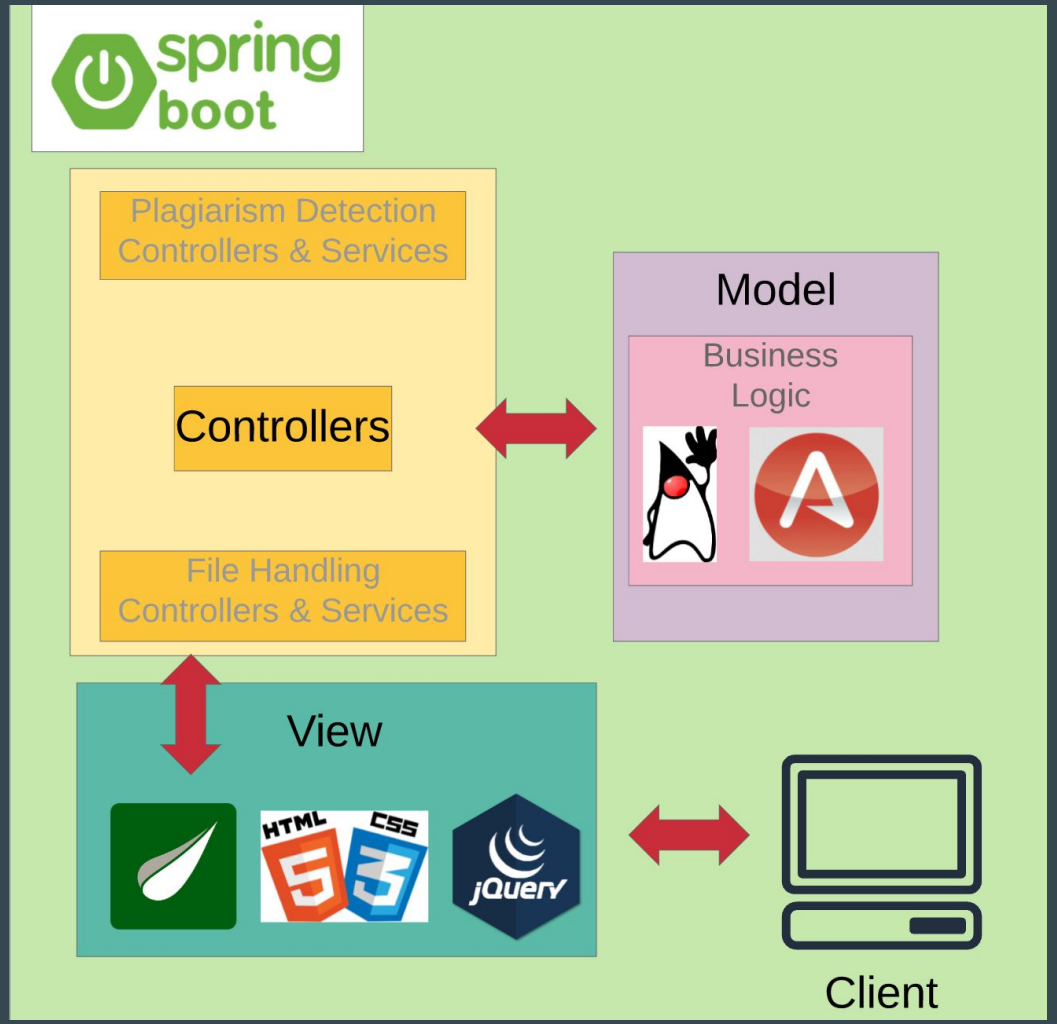
CS 5500 - Section 1
Foundations of Software Engineering
Dr. Frank Tip
Fall 2019

Khoury College of Computer Sciences
Northeastern University, Boston, MA

Proposed System Architecture ~ version 0.0.1

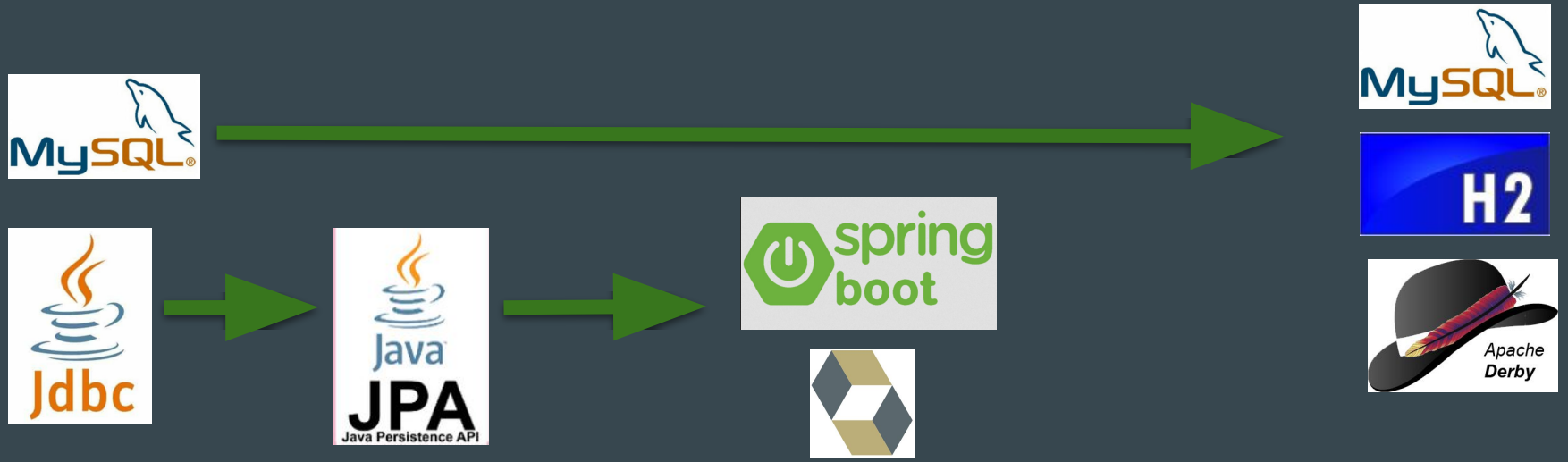


Current System Architecture ~ version 0.0.1



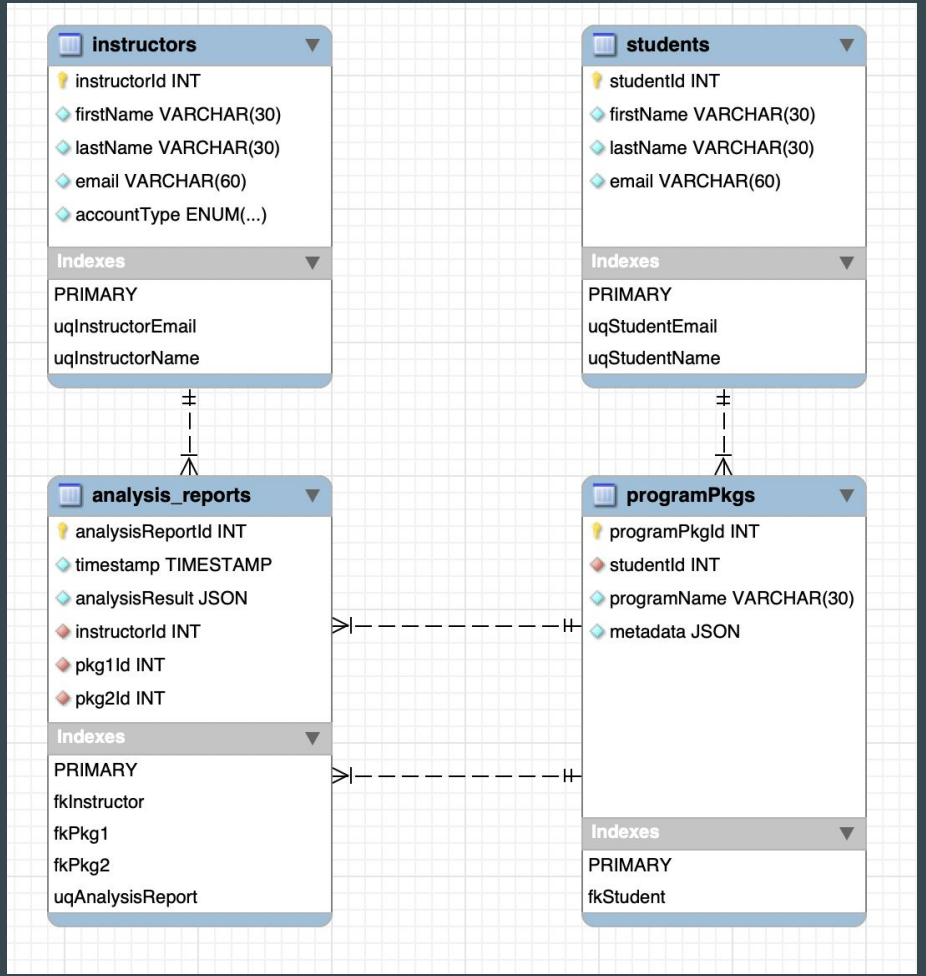
Architecture Update: Database Functionality

- Original proposal considered use cases for:
  - Instructor system registration and login
  - Student registration by an instructor
  - Storing of student files
  - Running of plagiarism detection analysis and storing of results



Architecture Update:
Database Functionality

MySQL UML



Architecture Update:
Database Functionality

MySQL UML → SQL DDL

```sql
DROP TABLE IF EXISTS instructors;
CREATE TABLE instructors
(
    instructorId                INT         NOT NULL AUTO_INCREMENT,
    firstName                   VARCHAR(30)  NOT NULL,
    lastName                    VARCHAR(30)  NOT NULL,
    email                       VARCHAR(320) NOT NULL,
    password                    VARCHAR(60)  NOT NULL,
    accountType                 ENUM('PROFESSOR','TEACHING_ASSISTANT', 'STAFF', 'OTHER') NOT NULL,
    PRIMARY KEY (instructorId),
    CONSTRAINT uqInstructorEmail    UNIQUE (email),
    CONSTRAINT uqInstructorName    UNIQUE (firstName, lastName)
);

DROP TABLE IF EXISTS students;
CREATE TABLE students
(
    studentId                   INT         NOT NULL AUTO_INCREMENT,
    firstName                   VARCHAR(30)  NOT NULL,
    lastName                    VARCHAR(30)  NOT NULL,
    email                       VARCHAR(320) NOT NULL,
    PRIMARY KEY (studentId),
    CONSTRAINT uqStudentEmail       UNIQUE (email),
    CONSTRAINT uqStudentName        UNIQUE (firstName, lastName)
);

DROP TABLE IF EXISTS programPkgs;
CREATE TABLE programPkgs
(
    programPkgId                        INT      NOT NULL AUTO_INCREMENT,
    studentId                   INT         NOT NULL,
    programName                 VARCHAR(30)  NOT NULL,
    file                        BLOB         NOT NULL,
    metadata                    JSON         NOT NULL,
    PRIMARY KEY (programPkgId),
    CONSTRAINT fkStudent    FOREIGN KEY (studentId)       REFERENCES students(studentId)
);
```

Architecture Update:
Database Functionality

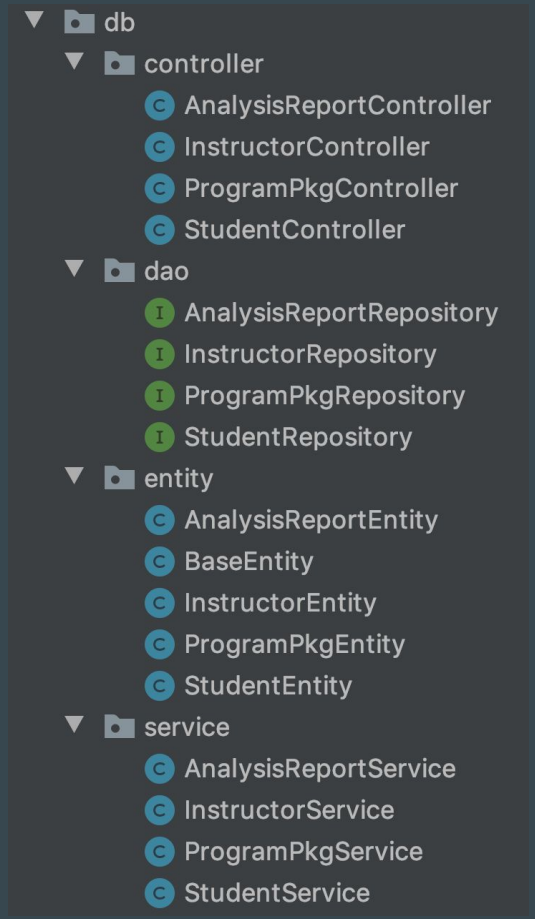
MySQL UML → SQL DDL → Spring-Data JPA
Hibernate
MySQL Connector

- JPA - Java Persistence API
  - Spring Data JPA improves the implementation of data access layers by abstracting SQL CRUD operations.
  - Tables are mapped to objects and provided supported query operations.
- Hibernate Object Relational Mapping Tool

```java
@Entity
@Table(name = "students",
        uniqueConstraints = {
                @UniqueConstraint(columnNames = {"firstName", "lastName"}),
                @UniqueConstraint(columnNames = {"email"})})

public class StudentEntity implements Serializable {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  @Column(name = "studentId")
  private Integer studentId;

  @NotNull
  @Email
  @Size(max = 320)
  @Column(name = "email", unique = true)
  private String email;

  @NotNull
  @Size(max = 30)
  @Column(name = "firstName")
  private String firstName;

  @NotNull
  @Size(max = 30)
  @Column(name = "lastName")
  private String lastName;

  @OneToMany(
          mappedBy = "programPkgs",
          cascade = CascadeType.ALL
  )
  private List<ProgramPkgEntity> programPkgEntityList = new ArrayList<ProgramPkgEntity>();

  public StudentEntity() {
  }

  public StudentEntity(Integer studentId, String email, String firstName, String lastName) {
    this.studentId = studentId;
    this.email = email;
    this.firstName = firstName;
    this.lastName = lastName;
  }
```

Architecture Update:
Database Functionality

- The majority of the entities and base CRUD operation methods have been implemented.
- Having never worked with REST controllers, Spring web framework or web development tools before, the learning curve was very steep.
- Difficult to unit test using mocks with MySQL, H2, Apache Derby
- Team re-evaluated deliverables and emphasis shifted to producing a minimum viable product for demo of algorithm.
- Database use cases have been migrated to a development branch.
- DB functionality can be implemented in version 0.0.2

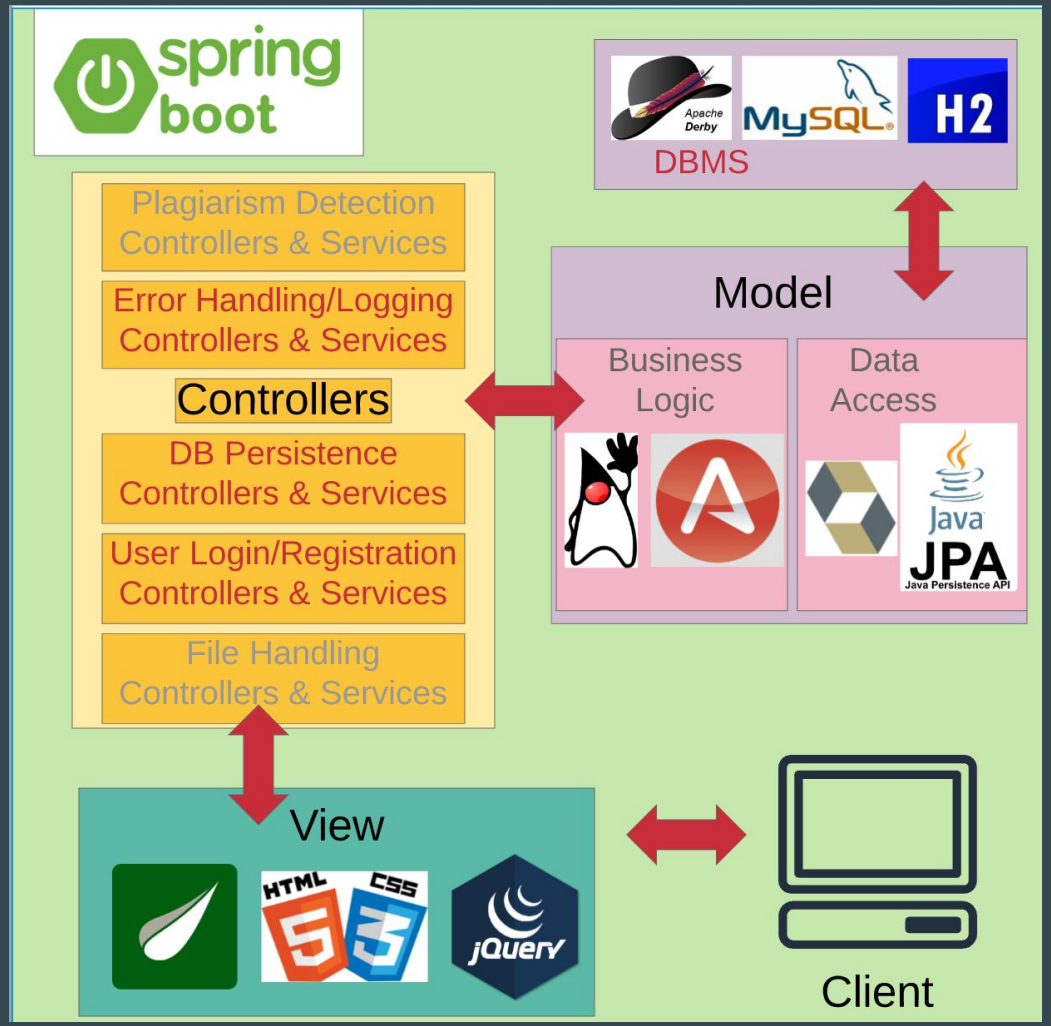


Architecture Update: Spring Boot/View/Server

- Spring MVC
- Apache Tomcat
- Java Server Pages (JSP)

- Spring Boot
- Spring-Boot-Starter-Web (Spring MVC)
- Spring-Boot-Starter-Thymeleaf
- jQuery

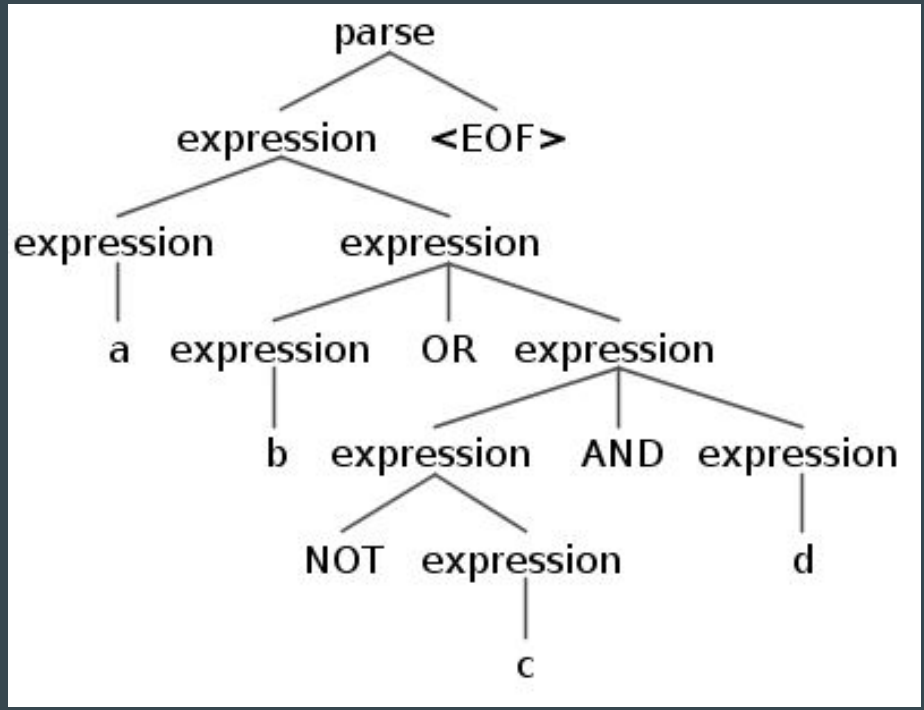
Future*
System Architecture
~ version 0.0.2



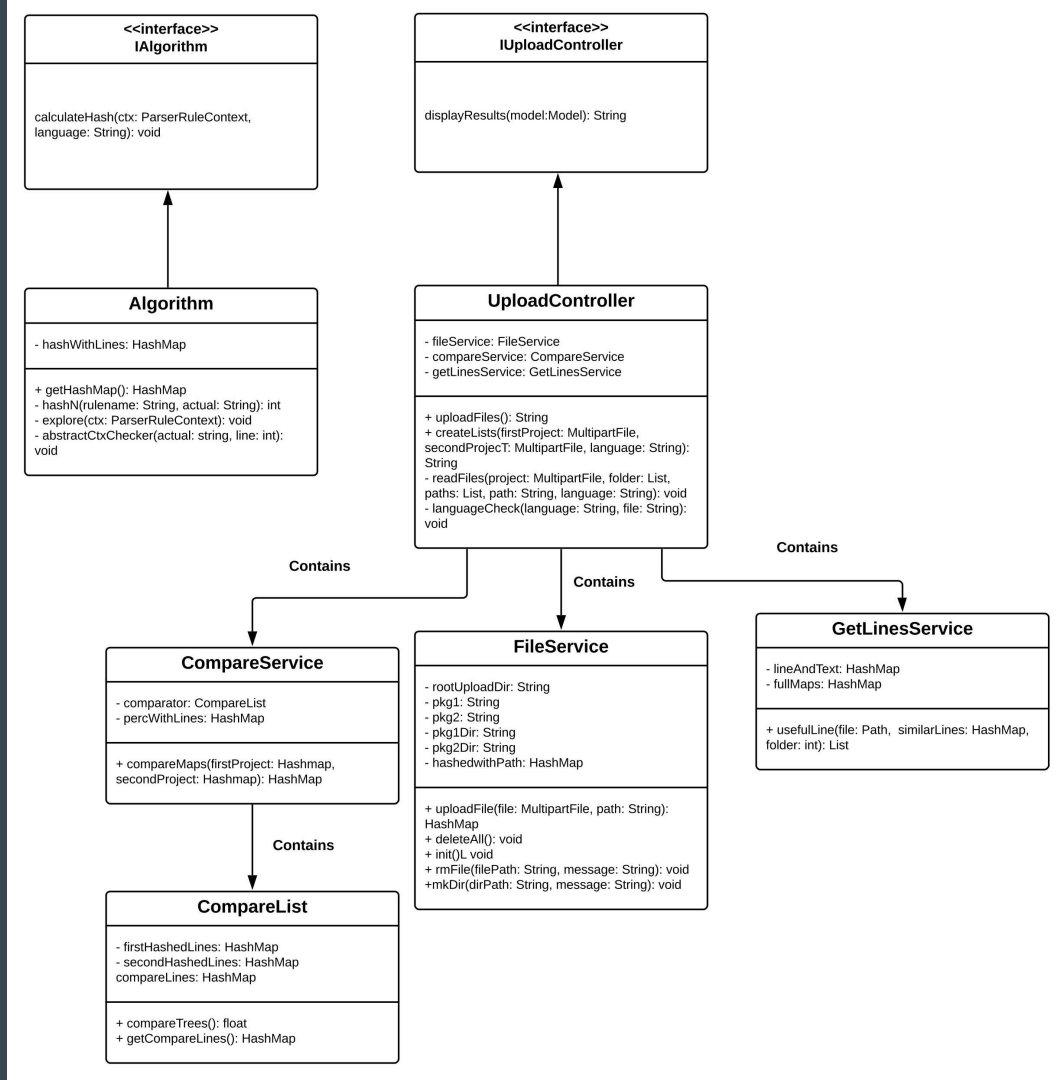
* sometime after the course concludes

Data Structure

- Data is parsed using ANTLR (ANother Tool for Language Recognition)
  - open source software for parsing code
- Data is stored in ParseTree object
  - Class created by ANTLR
- AST has methods to navigate tree and get data



UML Class Diagram



Algorithm

- Document Fingerprint
  - Based on Winnowing algorithm
  - Instead of using k-grams, each line is stored as a gram
- Recursively iterate through each AST
- Hash node content and add to HashMap
  - Store line number of node
- Compare all Hashmaps from Package 1 to all Hashmaps from Package 2
- Compare hash values and save similar lines

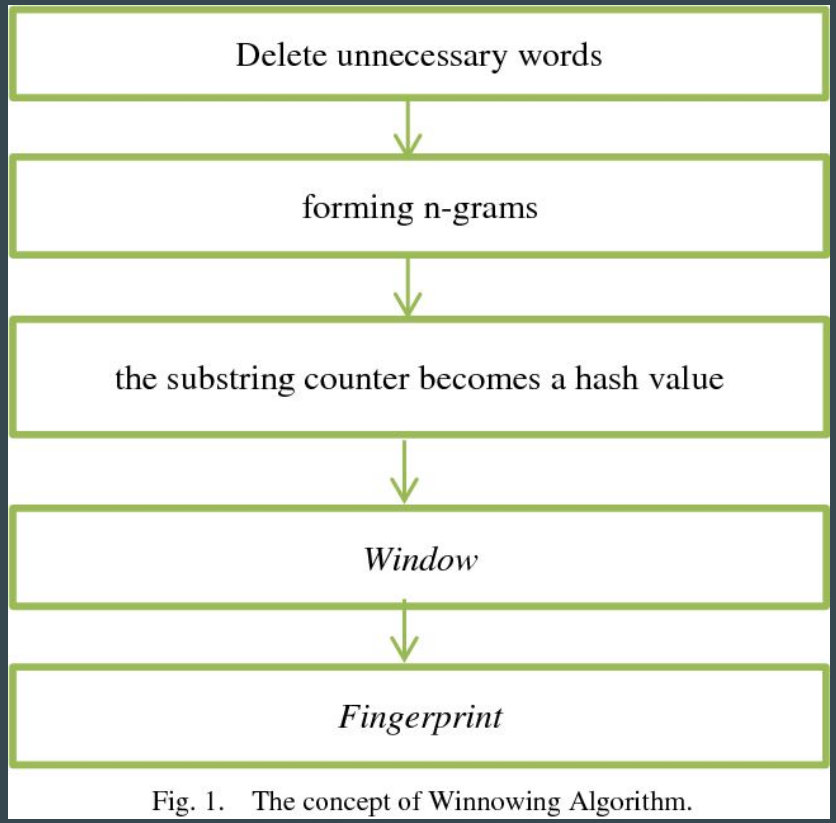


Fig. 1. The concept of Winnowing Algorithm.

Resources

https://www.semanticscholar.org/paper/Implementation-of-Winnowing-Algorithm-with-to-Yudhana-Sunardi/f4d33da8af7364d372af1f3f05c3dfc6c35a8ee0/figure/0

DEMO

Questions?