Abstract

We design and implement the first component of a group of programs that provide assistance for instructors who must evaluate student assignments. Many have used systems to accomplish portions of the task described; here we design and implement the submittal component of a comprehensive system. This system consists of a submittal component, which collects students’ code and/or other deliverables and stores them in a protected, uniform format, and a testing component, which processes each assignment in a way appropriate to the type, for example, compiling and running each programming assignment against a suite of test cases.
## Contents

1 Introduction ................................................. 4  
   1.1 Project Background and Need ............................... 4  
   1.2 Project Function ......................................... 5  

2 Design Specification ........................................... 6  
   2.1 Architecture Overview .................................... 6  
   2.2 Module / Object Specification .............................. 8  
   2.3 User Interface Specification ................................. 9  
      2.3.1 Login ............................................ 9  
      2.3.2 Welcome .......................................... 10  
      2.3.3 Course Listing .................................... 11  
      2.3.4 Course Administration (Instructors) .................... 12  
      2.3.5 Course Index – Inactive (Students) .................... 13  
      2.3.6 Course Index – Active (Students) ..................... 14  
      2.3.7 Assignment Administration (Instructors) ............... 15  
      2.3.8 Assignment Index (Students) ........................ 16  
      2.3.9 System Administration ................................ 17  

3 Implementation ............................................... 18  
   3.1 Database Server ........................................ 18  
      3.1.1 Schema .......................................... 18  
      3.1.2 Initial Values ..................................... 22  
   3.2 Database / Web Page Communication Layer ................. 24  
   3.3 Dynamic Web Pages ...................................... 24  
   3.4 Submission Archives .................................... 24
1 Introduction

The Checkmate system will provide a uniform and consistent method for students to turn in their programming assignments and then test students’ computer programs for correctness by running each program automatically against the instructor’s test cases. This allows more accurate, thorough, and reliable evaluation than conventional methods of grading.

The system will consist of both a submittal component and a testing component. The submittal component will be used by the student whenever he or she completes an assignment. It will store the assignment in a location accessible solely to the instructor with a timestamp. The testing component will run each submitted assignment against a set of test cases provided by the instructor. It will then provide the results of these test runs to the instructor for comparison against known correct results.

This document outlines the background, design, and implementation of the submittal component of the Checkmate system. It is divided into three main sections. The Introduction gives an overview of the Checkmate system. The Design Specification provides an in-depth explanation of the design, architecture, modules, and user interface of the system. Finally the Implementation section contains a listing of the source code in its entirety.

1.1 Project Background and Need

Even though methods for submitting student assignments electronically already exist, they are prone to error and inconsistency\(^1\), resulting in both frustration on the student’s and instructor’s part. Moreover, since the assignment submittals are inconsistent, it is

---

\(^1\)At UC Irvine in the Department of Information & Computer Science, the method for submitting assignments electronically is to upload a folder containing all the required components to a specific network share (dropbox). The student is required to name his/her folder consistently and to ensure that everything the instructor wants is there. This currently results in folders being named inconsistently, incomplete or incorrect submittals, and submittals to the wrong dropbox.
impossible to automatically run tests against the assignment to determine correctness, plagiarism, etc.

1.2 Project Function

Because of these problems with current methods, a new method must be devised. The Checkmate submittal component fills this need. It provides a method for instructors to define assignments for the courses they teach, stating every part of the assignment they want handed in, and the file type and quantity of that part. This prevents the student from handing in superfluous or incorrect items. Additionally, those items uploaded are filed by course, then by student. This consistent labeling ensures there is never confusion over to whom a submission belongs, or for what course it is. Finally, since the location and content of submissions is well-defined, it provides a perfect environment for an automatic grading program of any type to operate within. Thus, the Checkmate submittal component corrects the problems of current systems which put the burden of correct submittal on the student.
2 Design Specification

2.1 Architecture Overview

The basic architecture of the Checkmate submittal component is the Model-View-Controller architecture.

The Model contains the JavaBeans and Database Server. All outside communication to and from the Controller happens via this mechanism. The beans are responsible for
retrieving records from the database when they are needed and updating or storing new records. The beans use JDBC to communicate to and from the database.

The View is generated by JavaServer Pages (JSP) or Servlets, based on the information given by the beans. It is then sent to the web browser where it is rendered for the user. The web browser then takes the function of Controller, allowing the user to interact with the HTML form and sending the user’s request or changes back to the JSP or Servlet. The changes or request is then forwarded to the beans which take the corresponding action with the database server, completing the circle.

The database schema is modeled here using an Entity-Relationship (ER) diagram.
2.2 Module / Object Specification

Please see Checkmate JavaDoc for the Module / Object specification.
2.3 User Interface Specification

(Editor’s note: this section needs to be updated.)

2.3.1 Login

The first thing the user must do to be able to use the Checkmate submittal component is identify him/herself to the system. This is accomplished by a login page first being presented to the user.
2.3.2 Welcome

After the user logs in successfully, he/she is presented with this “Welcome” page. From there, the user can choose a course he/she is related to from the button bar at the top, or switch periods to work within a different period. The user can also select “Course Listing” to view a list of all the Checkmate courses for that period. If the user has “root” privileges, he/she can also access the “System Administration” page.
2.3.3 Course Listing

The course listing shows every course in the selected period that has been registered with Checkmate. When the user clicks the “Go” button next to the course information, and he/she is not already affiliated with the course, he/she will be presented with the “Course Index – Inactive” page. If he/she is affiliated as a student, he/she will be presented with the “Course Index – Active” page. If he/she is affiliated as an instructor, he/she will be presented with the “Course Administration” page.
2.3.4 Course Administration (Instructors)

The course administration page allows an instructor to add or remove instructors from the course, as well as to change the assignment list. The instructor can modify or remove already existing assignments or add new assignments to the list. Assignments on this list will be available to affiliated students for submittal.
2.3.5 Course Index – Inactive (Students)

The course index – inactive page presents an unaffiliated user with one option: to activate him/herself with the course. This will affiliate the user with the course as a student and present him/her with the “Course Index – Active” page.
2.3.6 Course Index – Active (Students)

From the course index – active page, students can choose to unaffiliate themselves with the course by clicking the “Drop” button, or to see the details of or submit an assignment by clicking the “Details / Submit” button.
2.3.7 Assignment Administration (Instructors)

The assignment administration page allows an instructor for the course to set the assignment’s number (alphanumeric), the assignment’s name, and when the assignment is due. He/she can also add, modify, or remove parts of the assignment. An assignment must have at least one part for the student to be able to submit files for it.
The assignment index page is the students’ view of the instructors’ assignment administration page. Here the student is shown the parts of the assignment which he/she has already submitted and given the opportunity to remove those parts. The student can also submit files for each part.
2.3.9 System Administration

The system administration page is only accessible by the “root” user, ID 0. From the system administration page root can add courses, assignment part (file) types, or users.
3 Implementation

3.1 Database Server

The database server chosen for implementation is PostgreSQL\(^2\), an open-source object relational database, because of its support for JDBC and SQL.

3.1.1 Schema

The SQL statements used to create the schema are as follows:

```sql
CREATE TABLE users
(
    user_id varchar(9) not null primary key,
    user_email varchar(40) not null,
    user_realname varchar(40) not null,
    user_password varchar(15),
    current_year char(4),
    current_quarter char
);

CREATE TABLE valid_course_quarters
(
    course_quarter char not null primary key,
    course_quarter_name varchar(15) not null,
    course_quarter_sort integer not null
);

CREATE TABLE valid_course_departments
(
    course_department varchar(20) not null primary key
);

CREATE TABLE courses
(
    course_code char(5) not null,
    course_year char(4) not null CHECK (course_year ~ '^2[0-9][0-9][0-9]$'),
    course_quarter char not null,
    course_department varchar(20) not null,
    course_number varchar(5) not null,
    primary key( course_code, course_year, course_quarter ),
    CONSTRAINT quarter_exists
        FOREIGN KEY(course_quarter) REFERENCES valid_course_quarters
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    CONSTRAINT department_exists
        FOREIGN KEY(course_department) REFERENCES valid_course_departments
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

\(^2\)For more information, visit the PostgreSQL website: http://www.postgresql.org/
CREATE TABLE valid_roles
(
    role_name varchar(10) not null primary key
);

CREATE TABLE roles
(
    user_id varchar(9) not null,
    course_code char(5) not null,
    course_year char(4) not null,
    course_quarter char not null,
    role_name varchar(10) not null,
    primary key(user_id, course_code, course_year, course_quarter),
    CONSTRAINT user_exists
        FOREIGN KEY(user_id) REFERENCES users
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT course_exists
        FOREIGN KEY(course_code, course_year, course_quarter) REFERENCES courses
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT role_exists
        FOREIGN KEY(role_name) REFERENCES valid_roles
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);

CREATE TABLE assignments
(
    course_code char(5) not null,
    course_year char(4) not null,
    course_quarter char not null,
    assignment_number varchar(10) not null,
    assignment_name varchar(60) not null,
    assignment_due timestamp with time zone,
    assignment_cutoff timestamp with time zone
        CHECK (assignment_cutoff >= assignment_due),
    assignment_shown timestamp with time zone,
    assignment_hidden timestamp with time zone
        CHECK (assignment_hidden >= assignment_shown),
    primary key(course_code, course_year, course_quarter, assignment_number),
    CONSTRAINT course_exists
        FOREIGN KEY(course_code, course_year, course_quarter) REFERENCES courses
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE assignment_parts
(
    course_code char(5) not null,
    course_year char(4) not null,
    course_quarter char not null,
    assignment_number varchar(10) not null,
    assignment_part_number varchar(10) not null,
    assignment_part_name varchar(60) not null,
    assignment_part_filename varchar(80) not null,
    assignment_part_quantity_min integer not null,
    assignment_part_quantity_max integer not null,
    assignment_part_description text,
    primary key(course_code, course_year, course_quarter, assignment_number,
                 assignment_part_number),
    CONSTRAINT course_assignment_exists
    FOREIGN KEY(course_code, course_year, course_quarter, assignment_number)
    REFERENCES assignments
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE submissions
(
    user_id varchar(9) not null,
    course_code char(5) not null,
    course_year char(4) not null,
    course_quarter char not null,
    assignment_number varchar(10) not null,
    assignment_part_number varchar(10) not null,
    submission_filename varchar(40) not null,
    submission_timestamp timestamp with time zone not null,
    submission_blob oid,
    removal_timestamp timestamp with time zone,
    PRIMARY KEY (user_id, course_code, course_year,
                 course_quarter, assignment_number, assignment_part_number,
                 submission_filename, submission_timestamp),
    CONSTRAINT user_exists
    FOREIGN KEY(user_id) REFERENCES users
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    CONSTRAINT course_assignment_part_exists
    FOREIGN KEY(course_code, course_year, course_quarter, assignment_number,
                 assignment_part_number)
    REFERENCES assignment_parts
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE UNIQUE INDEX unique_submission_filename ON submissions
(user_id, course_code, course_year, course_quarter, assignment_number,
 assignment_part_number, submission_filename)
WHERE removal_timestamp IS NULL;
CREATE TABLE user_logins
(
    user_id varchar(9) not null,
    login_timestamp timestamp with time zone not null,
    login_from text not null,
    login_successful boolean,
    primary key(user_id, login_timestamp),
    CONSTRAINT user_exists
        FOREIGN KEY(user_id) REFERENCES users
            ON DELETE CASCADE
            ON UPDATE CASCADE
);

CREATE FUNCTION is_assignment_part_complete(bigint, integer, integer)
    RETURNS boolean
    AS 'SELECT CASE WHEN $2 = 0 THEN TRUE ELSE $1 >= $2 END AS RESULT'
        LANGUAGE 'sql';

CREATE FUNCTION boolean_and(boolean, boolean)
    RETURNS boolean
    AS 'SELECT $1 AND $2 AS RESULT'
        LANGUAGE 'sql';

CREATE AGGREGATE all_true
(
    sfunc1 = boolean_and,
    basetype = boolean,
    stype1 = boolean,
    initcond1 = 'true'
);

CREATE TABLE sections
(
    course_code char(5) not null,
    course_year char(4) not null,
    course_quarter char not null,
    section_id varchar(10) not null,
    section_name varchar(60) not null,
    primary key( course_code, course_year, course_quarter, section_id ),
    CONSTRAINT course_exists
        FOREIGN KEY(course_code, course_year, course_quarter) REFERENCES courses
            ON DELETE CASCADE
            ON UPDATE CASCADE
);

CREATE TABLE section_membership
(
    course_code char(5) not null,
    course_year char(4) not null,
    course_quarter char not null,
    section_id varchar(10) not null,
    user_id varchar(9) not null,
    primary key(course_code, course_year, course_quarter, section_id, user_id),
    CONSTRAINT section_exists
        FOREIGN KEY(course_code, course_year, course_quarter, section_id, user_id)
          REFERENCES sections
            ON DELETE CASCADE
            ON UPDATE CASCADE,
    CONSTRAINT user_exists
        FOREIGN KEY(user_id) REFERENCES users
            ON DELETE CASCADE
            ON UPDATE CASCADE
);

CREATE TABLE global_roles
(
    user_id varchar(9) not null,
    role_name varchar(10) not null,
    primary key(user_id, role_name),
    CONSTRAINT user_exists
        FOREIGN KEY(user_id) REFERENCES users
            ON DELETE CASCADE
            ON UPDATE CASCADE,
    CONSTRAINT role_exists
        FOREIGN KEY(role_name) REFERENCES valid_roles
            ON DELETE RESTRICT
            ON UPDATE CASCADE
);

3.1.2 Initial Values

The database needs to be seeded with the following values.

INSERT INTO valid_roles VALUES ('Student');
INSERT INTO valid_roles VALUES ('Instructor');
INSERT INTO valid_roles VALUES ('Assistant');
INSERT INTO valid_roles VALUES ('Sys Admin');

INSERT INTO users VALUES ('root', 'administrator@institution.tld', 'Super-User', 'encrypted_password');

The database also needs to be seeded with the following data for use at UC Irvine.

INSERT INTO valid_course_quarters VALUES ('F', 'Fall', 6);
INSERT INTO valid_course_quarters VALUES ('M', 'Summer 10wk', 5);
INSERT INTO valid_course_quarters VALUES ('Z', 'Summer 2', 4);
INSERT INTO valid_course_quarters VALUES ('S', 'Summer 1', 3);
INSERT INTO valid_course_quarters VALUES ('W', 'Winter', 1);

INSERT INTO valid_course.departments VALUES ('AFAM');
INSERT INTO valid_course.departments VALUES ('ANATOMY');
INSERT INTO valid_course.departments VALUES ('ANESTH');
INSERT INTO valid_course.departments VALUES ('ANTHRO');
INSERT INTO valid_course.departments VALUES ('ART HIS');
INSERT INTO valid_course.departments VALUES ('ART STU');
INSERT INTO valid_course.departments VALUES ('ARTS');
INSERT INTO valid_course.departments VALUES ('ASIANAM');
INSERT INTO valid_course.departments VALUES ('BIO SCI');
INSERT INTO valid_course.departments VALUES ('BIOCHEM');
INSERT INTO valid_course.departments VALUES ('CAMPREC');
INSERT INTO valid_course.departments VALUES ('CBEMS');
INSERT INTO valid_course.departments VALUES ('CEM');
INSERT INTO valid_course.departments VALUES ('CHEM');
INSERT INTO valid_course.departments VALUES ('CHINESE');
INSERT INTO valid_course.departments VALUES ('CLASSIC');
INSERT INTO valid_course.departments VALUES ('COM LIT');
INSERT INTO valid_course.departments VALUES ('CRITISM');
INSERT INTO valid_course.departments VALUES ('CRM/LAW');
INSERT INTO valid_course.departments VALUES ('DANCE');
INSERT INTO valid_course.departments VALUES ('DERM');
INSERT INTO valid_course.departments VALUES ('DEV BIO');
INSERT INTO valid_course.departments VALUES ('DRAMA');
INSERT INTO valid_course.departments VALUES ('E ASIAN');
INSERT INTO valid_course_departments VALUES('EARTHSS');
INSERT INTO valid_course_departments VALUES('ECO EVO');
INSERT INTO valid_course_departments VALUES('ECON');
INSERT INTO valid_course_departments VALUES('ED AFF');
INSERT INTO valid_course_departments VALUES('EDU AFF');
INSERT INTO valid_course_departments VALUES('EDU ABR');
INSERT INTO valid_course_departments VALUES('EDUC');
INSERT INTO valid_course_departments VALUES('ENGLISH');
INSERT INTO valid_course_departments VALUES('ENGR');
INSERT INTO valid_course_departments VALUES('ENGR AE');
INSERT INTO valid_course_departments VALUES('ENGR BE');
INSERT INTO valid_course_departments VALUES('ENGR CE');
INSERT INTO valid_course_departments VALUES('ENGR ME');
INSERT INTO valid_course_departments VALUES('ENGRCBE');
INSERT INTO valid_course_departments VALUES('ENGRCEEE');
INSERT INTO valid_course_departments VALUES('ENGRCHE');
INSERT INTO valid_course_departments VALUES('ENGRECE');
INSERT INTO valid_course_departments VALUES('ENGRMAE');
INSERT INTO valid_course_departments VALUES('ENGRMSE');
INSERT INTO valid_course_departments VALUES('ENVI RON');
INSERT INTO valid_course_departments VALUES('FAM MED');
INSERT INTO valid_course_departments VALUES('FILMSTD');
INSERT INTO valid_course_departments VALUES('FINEART');
INSERT INTO valid_course_departments VALUES('FRENCH');
INSERT INTO valid_course_departments VALUES('GERMAN');
INSERT INTO valid_course_departments VALUES('GREEK');
INSERT INTO valid_course_departments VALUES('HISTORY');
INSERT INTO valid_course_departments VALUES('HUMAN');
INSERT INTO valid_course_departments VALUES('I&C SCI');
INSERT INTO valid_course_departments VALUES('IN4MATX');
INSERT INTO valid_course_departments VALUES('INT M ED');
INSERT INTO valid_course_departments VALUES('INTL ST');
INSERT INTO valid_course_departments VALUES('ITALIAN');
INSERT INTO valid_course_departments VALUES('JAPANSE');
INSERT INTO valid_course_departments VALUES('JUDAI CA');
INSERT INTO valid_course_departments VALUES('KOREAN');
INSERT INTO valid_course_departments VALUES('LATIN');
INSERT INTO valid_course_departments VALUES('LINGUIS');
INSERT INTO valid_course_departments VALUES('LFS');
INSERT INTO valid_course_departments VALUES('M&MG');
INSERT INTO valid_course_departments VALUES('MATH');
INSERT INTO valid_course_departments VALUES('MED ED');
INSERT INTO valid_course_departments VALUES('MGMT');
INSERT INTO valid_course_departments VALUES('MGMT EP');
INSERT INTO valid_course_departments VALUES('MGMT FE');
INSERT INTO valid_course_departments VALUES('MGMT HC');
INSERT INTO valid_course_departments VALUES('MIC BIO');
INSERT INTO valid_course_departments VALUES('MOL BIO');
INSERT INTO valid_course_departments VALUES('MUSIC');
INSERT INTO valid_course_departments VALUES('NEU RBIO');
INSERT INTO valid_course_departments VALUES('NEUROL');
INSERT INTO valid_course_departments VALUES('OB/GYN');
INSERT INTO valid_course_departments VALUES('OPHTHAL');
INSERT INTO valid_course_departments VALUES('PATH');
INSERT INTO valid_course_departments VALUES('PED GEN');
INSERT INTO valid_course_departments VALUES('Peds');
INSERT INTO valid_course_departments VALUES('PHARM');
INSERT INTO valid_course_departments VALUES('PHILOGS');
INSERT INTO valid_course_departments VALUES('PHY SCI');
INSERT INTO valid_course_departments VALUES('PHYSICS');
INSERT INTO valid_course_departments VALUES('PHYSIO');
INSERT INTO valid_course_departments VALUES('PM&R');
INSERT INTO valid_course_departments VALUES('POL SCI');
INSERT INTO valid_course_departments VALUES('PORTUG');
3.2 Database / Web Page Communication Layer

JavaBeans\(^3\) using JDBC were chosen as the glue between the database server and the dynamic web pages. JSP pages can seamlessly couple with JavaBeans, allowing for persistent beans across an entire session or just a particular request, and for fields to be automatically populated from form posts.

3.3 Dynamic Web Pages

JavaServer Pages (JSP)\(^4\) and Servlets\(^5\) were chosen as the technologies to use for generating dynamic web pages. Tomcat\(^6\) was chosen as the server for these since it is the definitive reference implementation for those two technologies.

3.4 Submission Archives

Student submissions are stored in the submissions table, as a BLOB. When they are accessed through the Checkmate system, they are extracted from the database and placed into a ZIP archive for downloading. Even though storing them outside of the

\(^{3}\)For more information, visit the JavaBeans website: http://java.sun.com/products/javabeans/

\(^{4}\)For more information, visit the JSP website: http://java.sun.com/products/jsp/

\(^{5}\)For more information, visit the Servlet website: http://java.sun.com/products/servlet/

\(^{6}\)For more information, visit the Tomcat website: http://jakarta.apache.org/tomcat/
database would allow for greater ease of accessing them outside of the Checkmate system, we chose to place them inside the database to ensure consistency between submission metadata and the actual submitted files.