CSE4101

# FIT Schedule Planner

Software Design Document

Pedro Moura - pmoura2020@my.fit.edu
Jordan Synodis - jsynodis2021@my.fit.edu
Dr. Fitz - fnembhard@fit.edu

Date: (09/30/2024)

# TABLE OF CONTENTS

Software Design Document

# 1. INTRODUCTION

## 1.1. Purpose

This Design Document displays the architecture and system design of the FIT Schedule Planner.

## 1.2. Scope

FIT Schedule Planner is a web application which comes with the intention to streamline the class registration process for students at Florida Institute of Technology (FIT). The goals are to reduce confusion during registration, prevent students from taking unnecessary courses, and assist in schedule planning by providing tools like conflict detection and advanced search filters. This can be possible by integrating various systems (such as course schedules, degree evaluations, and program requirements) into a single, user-friendly interface.

## 1.3. Overview

This document details the software design for the FIT Schedule Planner. It covers the system architecture, data design, component design, human interface design, and includes a requirements matrix.

## 1.4. Reference Material

FIT Course Schedules:

Fall Semester

Spring Semester

FIT Degree Programs:

Degree Requirements

Project Plan:

Project Plan Document

## 1.5. Definitions and Acronyms

**CAPP:** Curriculum, Advising, and Program Planning (Degree Evaluation System)

**PAWS:** Panther Access Web System (FIT's Student Portal)

# 2. SYSTEM OVERVIEW

FIT Schedule Planner is a web-based application with the goal to help FIT students in with planning their course schedules efficiently. It consolidates course listings, degree requirements, personal schedules, and professor ratings into one place. All that to minimize confusion and also streamline the registration process. Key features include:

- Viewing available classes and times
- Advanced search with multiple filters
- Schedule visualization with conflict detection
- Prerequisite and availability checking
- Degree progress tracking through uploaded CAPP Degree Evaluations
- Integration with RateMyProfessor for instructor ratings

# 3. SYSTEM ARCHITECTURE

## 3.1. Architectural Design

- User Interaction:
  - User interacts with the User Interface (UI) in the Presentation Layer.
- Presentation Layer:
  - Contains the UI, built with React.js.
- Application Layer:
  - API Gateway: Routes requests from the UI to appropriate modules.
  - Authentication Module:Manages user authentication using JWT.
  - Schedule Management Module: Handles schedule-related functionalities.
  - Degree Progress Module: Manages degree progress tracking.
  - Filter & Search Module: Provides advanced search capabilities.
- Data Layer:
  - MongoDB Database: Stores all application data.
- Data Flow:
  - External Data Sources (FIT Course Schedules & Programs) provide data via the Data Scraping Module, which updates the Database.
- Authentication Flow:
  - The Authentication Module issues and verifies JSON Web Tokens (JWT).
  - The UI sends JWTs with requests for authentication.
- System Architecture Diagram:

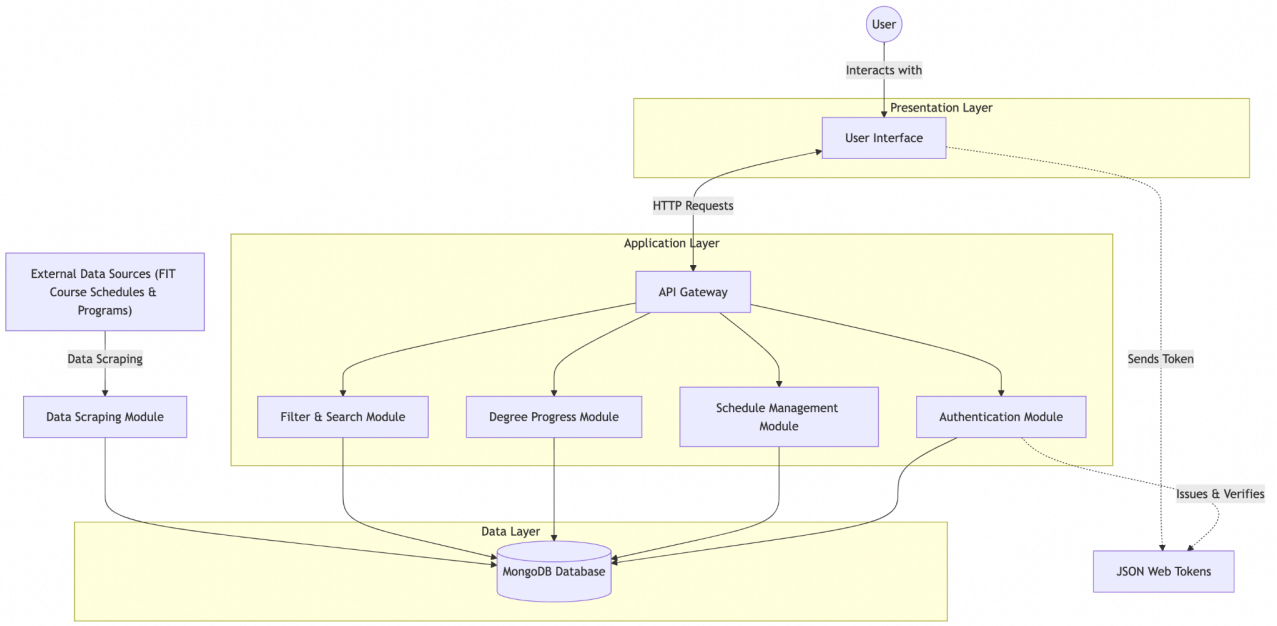



Fig. 1: UML of system architecture diagram

## 3.2. Design Rationale

The idea of our system is to be structured into distinct layers and modules. Dividing the system into the Presentation, Application, and Data layers allows each to be developed and maintained independently. This facilitates team collaboration and parallel development, and also make it modulable, scalable, and maintainable:

- **Modularity:** Each module in the Application Layer has a specific responsibility, making it easier to manage complexity and debug issues. Modules can be updated/replaced without significantly impacting other parts of the system.

- **Scalability:** The use of modular components allows the system to scale horizontally, making it possible to add more servers or nodes to distribute the load – if needed.

- **Technology Alignment:** React.js on the front-end and Node.js on the back-end allow for a unified development language (JavaScript), simplifying development processes. MongoDB's flexible schema accommodates the diverse data types and structures required by the application.

# 4. DATA DESIGN

## 4.1. Data Description

**Course Data:** Information about courses offered, scraped from FIT's course schedules.

**Degree Requirements:** Details of program requirements, scraped from FIT's degree programs.

**User Schedules:** Individual schedules created by users, including selected courses and time blocks.

**CAPP Degree Evaluations**: Uploaded by users to track degree progress.

Professor Ratings: Retrieved from RateMyProfessor to assist in course selection.

## 4.2. Data Dictionary

Entities:

- Course
  - Attributes:
    - course_number (String)
    - course_name (String)
    - credits (Integer)
    - professor (String)
    - schedule (List of TimeSlot)
    - prerequisites (List of Course)
    - capacity (Integer)
    - enrolled (Integer)
  - Methods:
    - is_full()

- UserSchedule
  - Attributes:
    - courses (List of Course)
    - time_blocks (List of TimeSlot)
  - Methods:
    - add_course(course)
    - remove_course(course)
    - check_conflicts(course)

- DegreeRequirement
  - Attributes:
    - program_name (String)
    - required_courses (List of Course)
  - Methods:
    - compare_with_capp(capp_evaluation)

- CAPP_Evaluation
  - Attributes:
    - completed_courses (List of Course)
- TimeSlot
  - Attributes:
    - day (String)
    - start_time (Time)
    - end_time (Time)

## 5. COMPONENT DESIGN

Function add_course(course) in UserSchedule:

```
Function add_course(course):
    if check_conflicts(course):
        Display "Time conflict detected."
        return False
    else if course.is_full():
        Display "Course is full."
        return False
    else if not prerequisites_met(course):
        Display "Prerequisites not met."
        return False
    else:
        courses.append(course)
        Update schedule visualization
        return True
```

Function check_conflicts(course) in UserSchedule:

```
Function check_conflicts(course):
    for existing_course in courses:
        if times_overlap(existing_course.schedule,
course.schedule):
            return True
    for block in time_blocks:
        if times_overlap(block, course.schedule):
            return True
    return False
```

Function prerequisites_met(course):

```
Function prerequisites_met(course):
    for prereq in course.prerequisites:
        if prereq not in CAPP_Evaluation.completed_courses:
            return False
    return True
```

# 6. HUMAN INTERFACE DESIGN

## 6.1. Overview of User Interface

**Navigation Bar:** Located at the top of the web app for easy access to different sections (Schedule Planner, Degree Checklist, FAQs, etc).

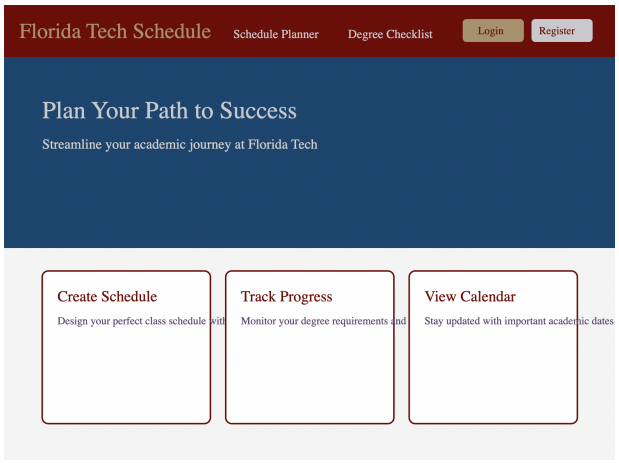
**Schedule Planner:** Allows users to search for classes using filters and visualize their weekly schedule.

**Degree Checklist:** Displays progress toward degree completion by comparing CAPP evaluations with program requirements.

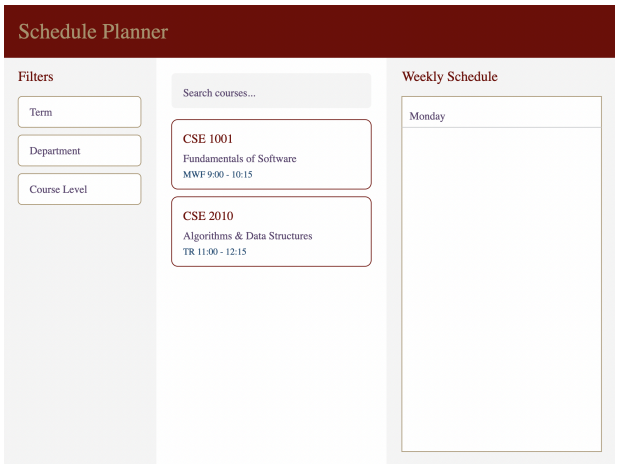
**Pop-up Notifications:** Inform users of errors, conflicts, or important updates.

## 6.2. Screen Images

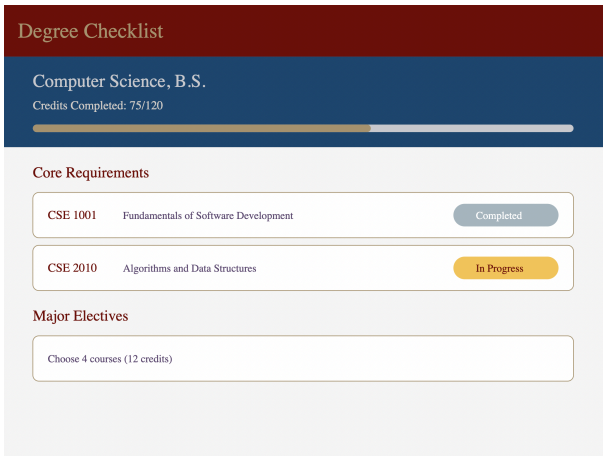
Home Page:




Schedule Planner Page:




Degree Checklist Page:



## 6.3. Screen Objects and Actions

**Filters:** Dropdowns and input fields for criteria like subject, professor, time slots, course number, credits, and ratings.

**Search Results:** List of courses matching the filters; clicking a course attempts to add it to the schedule.

**Schedule Grid:** Interactive weekly schedule where users can see added courses and detect time gaps.

**Degree Checklist Items:** Expandable items showing course details; completed courses are checked off.

**Navigation Links:** Direct access to different sections; consistent across all pages.

# 7. REQUIREMENTS MATRIX

| Task Requirement ID | Description | Components/Data Structures |
|---|---|---|
| TASK-1 | View available classes and times | Course, Data Scraping Module |
| TASK-2 | Add classes to schedule | UserSchedule, add_course() |
| TASK-3 | Visualize weekly schedule | User Interface Module |
| TASK-4 | Check prerequisites when adding classes | prerequisites_met(), CAPP_Evaluation |
| TASK-5 | Highlight overlapping classes | check_conflicts() |
| TASK-6 | Filter classes by criteria | Filter and Search Module |
| TASK-7 | Add personal time blocks | UserSchedule, time_blocks |
| TASK-8 | View degree progress checklist | DegreeProgress Module, DegreeRequirement |
| TASK-9 | Upload CAPP Degree Evaluation | DegreeProgress Module, CAPP_Evaluation |
| TASK-10 | Simple and consistent UI | User Interface Module |

Software Design Document