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:

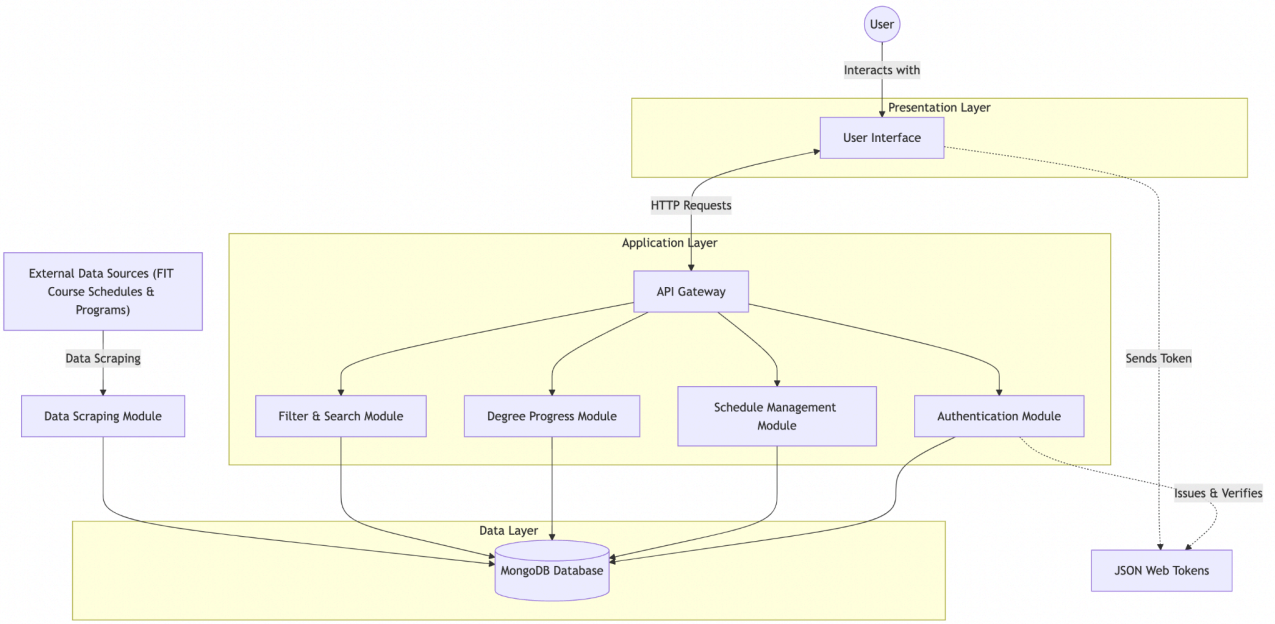


Fig. 1: UML of system architecture diagram

## 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 (//TODO)

## 4.1. Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

## 4.2. Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

# 5. COMPONENT DESIGN

In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

# 6. HUMAN INTERFACE DESIGN

## 6.1. Overview of User Interface

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

## 6.2. Screen Images

Display screenshots showing the interface from the user's perspective. These can be hand drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

## 6.3. Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

# 7. REQUIREMENTS MATRIX

Provide a crossreference that traces components and data structures to the requirements in your software requirements specification (SWRS) document.

Use a tabular format to show which system components satisfy each of the functional requirements from the SWRS. Refer to the functional requirements by the numbers/codes that you gave them in the SWRS.

# 8. APPENDICES

**9.**

*9.1. This section is optional.*

**10.**

**11.** Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.