G20 Summit

Project Portfolio

9/11/2023

[Introduction <Milestone 1: Proposal> 2](#_Toc125988975)

[The [Team Name] Team < Milestone 1: Proposal > 3](#_Toc125988976)

[System Requirements < Milestone 1: Proposal > 4](#_Toc125988977)

[Requirements [optional] 4](#_Toc125988978)

[Epics [optional] 4](#_Toc125988979)

[Epic #1 4](#_Toc125988980)

[User Stories 4](#_Toc125988981)

[User Story #1 4](#_Toc125988982)

[Project Management 5](#_Toc125988983)

[Continuity of Operations Plan (COOP) < Milestone 1: Proposal> 5](#_Toc125988984)

[Project Plan 0](#_Toc125988985)

[System Architecture Design and Development < Milestone 1: Proposal & Milestone 2: Architecture> 0](#_Toc125988986)

[System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation> 0](#_Toc125988987)

[Project Postmortem <Postmortem> 0](#_Toc125988988)

[Project Wins 0](#_Toc125988989)

[Root Cause Analysis 0](#_Toc125988990)

[Lessons Learned 0](#_Toc125988991)

[System Design <Milestone 2: System Architecture> 1](#_Toc125988992)

[System Architecture <Milestone 2: System Architecture> 1](#_Toc125988993)

[Component Design 1](#_Toc125988994)

[Data Flow 1](#_Toc125988995)

[System Components <Milestone 3: System Implementation> 1](#_Toc125988996)

[Component [Component Name 1] 1](#_Toc125988997)

[Component [Component Name 2] 1](#_Toc125988998)

[Component [Component Name n] 1](#_Toc125988999)

[Design Pattern <Milestone 3: System Implementation> 1](#_Toc125989000)

[Design Pattern <Milestone 3: System Implementation> 2](#_Toc125989001)

[System Implementation <Milestone 3: System Implementation> 3](#_Toc125989002)

[Project portfolio template directives and placeholders (delineated by “[ ]” or “< >” and/or highlighted or optional sections not included) should be removed from the document prior to submission. Empty sections for inclusion in later submissions may remain in the document for early submissions.]

[IMPORTANT: All diagrams developed using Enterprise Architectures must include the following acknowledgement: “Thanks to SPARX Systems for LSU student and faculty use of Enterprise Architect for academic purposes”.]

# Introduction

LSU currently uses a scheduling system and database that can be considered archaic for the modern web. The schedule booklet only sorts classes by the semester and department, forcing students to find their desired classes within a list of others that are most likely inconsequential for them. The obtuse scheduling system has lead to frustration within the student body due to the confusing layout, and has possibly resulted in some students not being able to choose the best classes for their specific needs. This updated scheduling system will solve the aforementioned issues by including a robust sorting system for all classes, a simple web interface that anyone can use, and an ability to view specific course and teacher information. This app will be made primarily with C++ with associated C++ data structures and libraries for the backend, along with front-end tools like HTML and javascript for the interface.

Core Features:

* Can filter and sort available classes based on multiple criteria
* An intuitive web interface that makes selecting classes simple
* Automatically updates available classes to choose based on class times already selected

Viable Features:

* Offers the ability to view the current or prior course syllabus for each class if applicable
* Denotes classes with special requirements such as a group project or a lab being involved
* Recommends classes based on the default course flowchart for each major

Stretch Features [optional]

* Estimate the quality of the class based on ratemyprofessor score
* Implement degree audit functionality

# The G20 Summit Team

Github link - https://github.com/hkaiserteaching/csc3380-fall-2023-project-group-20/tree/main

Preston Saxon – Team leader, organize meetings and progress checks

Alexander Leake – Backend development

Jacob Rogers – Backend development

Joel Rogers – Front-end development

Vivian San – Front-end development

# System Requirements

## Requirements

## Epics

## User Stories

### User Story #1

*As a student, I want to easily schedule and line up my classes for the semester, so I can worry less about navigating LSU’s course catalog.*

### User Story #2

*As a student, I want to know what the contents of the class I’m signing up for are, so I can be prepared for the classes I scheduled.*

# Project Management

## Continuity of Operations Plan (COOP)

In the event where a team member is sick and cannot fulfill their duties, their responsibilities regarding the project will be accounted for by the next member with the most time on their hands or at a later date once they are capable. If a team member cannot attend a meeting in-person or through zoom, they will be messaged about the content of the meetings and/or given a recording of the meeting. The absent member will still be in contact with the rest of the team concerning current developments on the project and any new requirements that must be met for their specific job.

## Project Plan

### System Architecture Design and Development

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| 1 | Constructing basic features and methods for the app |  | 70% |  | 9/20/23 | 10/7/23 |  |  |
| 1 | Making web interface for the app |  | 30% |  | 9/30/23 | 10/12/23 |  |  |
| 1 | Implementing syllabus functionality |  | 20% |  | 9/20/23 | 10/7/23 |  |  |

### System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation>

[Milestone 2 (Architecture): The Project Plan WBS provides a list of activities/tasks to be undertaken to complete Milestone 3 (System Implementation). The WBS activity chart should include task dependencies, estimated level of effort, and expected start and completion dates.

Milestone 3 (System Implementation): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
|  |  |  |  |  |  |  |  |  |

## Project Postmortem <Postmortem>

### Project Wins

[Provide a bulleted list of at least 3 positive aspects of the project.]

### Root Cause Analysis

[Provide a bulleted list of at least 3 negative aspects of the project. For each negative, provide the answer to the three successive “Why” questions. ]

### Lessons Learned

[For each negative aspect identified in the Root Cause Analysis, provide a mitigation strategy (i.e., what process should be introduced) to ensure that the problem is not repeated in subsequent projects.]

# System Design <Milestone 2: System Architecture>

The Class Scheduling System aims to assist students in choosing the optimal courses for each semester based on their degree requirements, completed courses, and course metadata. The system uses a central "Course Recommendations Engine" that interacts with multiple data stores to provide personalized course recommendations, taking into account factors like prerequisites, course availability, and workload.

## System Architecture <Milestone 2: System Architecture>

### The system architecture consists of a user interface for student interactions, a Course Recommendations Engine for generating personalized course lists, and four data stores for managing degree audits, completed courses, course catalogs, and course metadata. These components work in harmony to provide students with tailored course recommendations each semester.

### Component Design

Drawn using PlantUML by Joel Rogers

A diagram of a course degree

Description automatically generated

Data Flow Diagram

A diagram of a course

Description automatically generated Drawn by Joel Rogers using Draw.io

**Architecture Overview for Course Recommendation System**

*User I/O (Input/Output):*

Input Preferences Interface: Allows students to input their course-related preferences like workload, number of classes, and willingness to engage in group projects.  
Degree Audit Request Interface: Allows students to request their degree audit.  
Course Recommendation Interface: Displays the list of recommended courses to the student based on the gathered information.

External Data Sources:

Degree Audit Database: An external or integrated database storing all student degree audits, accessed to determine student's completed and required courses.  
Course Catalog Database: Another external or integrated database that holds information on all available courses, their prerequisites, and seasonal availability (fall/spring).  
Course Metadata Database: A third database storing metadata from class syllabi and external websites that can provide additional context like course difficulty, homework load, etc.

Major System Components:

Gather Degree Audit Component: Responsible for pulling relevant degree audit data from the Degree Audit Database.  
Gather Student Preferences Component: Collects and processes student preferences from the user interface.  
Course Recommendation Algorithm Component: Utilizes both the degree audit and student preferences to generate a tailored list of course recommendations.  
Metadata Gathering Component: Fetches and updates course metadata in the Course Metadata Database.

Data Flows and Interactions:

* Student inputs their preferences via the Input Preferences Interface which is processed by the Gather Student Preferences Component.
* Student requests degree audit through the Degree Audit Request Interface. Gather Degree Audit Component fetches this data from the Degree Audit Database.
* Course Recommendation Algorithm Component pulls data from both the Gather Degree Audit Component and Gather Student Preferences Component.
* Course Recommendation Algorithm Component also interacts with Course Catalog Database and Course Metadata Database for additional data needed for recommendations.
* The list of recommended courses is then displayed to the Student via the Course Recommendation Interface.

## System Components <Milestone 3: System Implementation>

[*Include a component sub-section for each component in the architecture diagram. Each component subsection will include a class diagram*]

### Component [Component Name 1]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

### Component [Component Name 2]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

### Component [Component Name n]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. The project must include at least design patterns covered in class. Include the name of the team member that created the diagram in EA.*]

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. Include the name of the team member that created the diagram in EA. A second design pattern may be included for bonus points.*]

# System Implementation <Milestone 3: System Implementation>

[*In the table below, include a row for each component in your System Architecture diagram. In the second column, list the programming language(s) used to implement the component and the what % of that programming language is used in the implementation. In the third column, list the team member(s) that implement the component and what % of that implementation was completed by that team member. IMPORTANT NOTE: All architectural components must be implemented by an object-oriented programming language: Java, C++, or C#.*]

|  |  |  |
| --- | --- | --- |
| **Architectural Component** | **Programming Language(s) %** | **Team Member(s) %** |
| *[Data Manager]* | *[C++ (45%)*  *Java (55%)]* | *[Mickey Mouse (15%)*  *Donald Duck (20%)*  *Daisy Duck (40%*  *Pluto (25%)]* |