Class Scheduler

Design Document

Team 1
ECpE Student Services
Brian Schomer (Meeting lead/minutes taker), Carter Everts (Testing lead), Isaiah John Ortiola(Ind. Component Designer), Lewis Callaway (Client Interaction), Michael Less (Tech Lead), Simeon Steward (Team organization)
Sdmay24-01@iastate.edu
https://sdmay24-01.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

Each Ticket will have a branch associated with it, named as such: #ticketid#_branch_title

Each branch shall be reviewed by one or more other people before being merged into main. The merge must be completed by someone other than the developer who worked on the branch.

Summary of Requirements

- Create a Desktop app that can find valid class times for new classes.
- App will minimize conflicts for students that the course is designed for.
- App will recommend timeslots so that user can choose what works best.
- App has a user interface that is simple to use.
- App must be able to input and store courses that are already scheduled.

Applicable Courses from Iowa State University Curriculum

- Computer Science: 228, 311, 309, 319, 363
- Software Engineering: 317
- Computer Engineering: 230

New Skills/Knowledge acquired that was not taught in courses

Scheduling Algorithms, Front-end Development

Table of Contents

Contents

Contents 2

1 Team 7

1.1 Team Members 7

1.2 Required Skill Sets for Your Project 7

Front-End Programming (specifically desktop applications), Backend Programming (including algorithms for scheduling/conflict graphs), communication with client, time management. 7

1.3 Skill Sets covered by the Team 7

All team members have gone through the same set of core classes and have an ability to troubleshoot and do full-stack development. This means there will be overlap with the skills. 7

Frontend: Michael Less 7

Backend: Simeon Steward, Brian Schomer 7

Testing: Carter Everts 7

Component Designer: Isaiah Ortiola 7

Client Communication: Lewis Callaway 7

Time Management: Simeon, Lewis, and Brian 7

1.4 Project Management Style Adopted by the team 7

We will follow the agile system with roles assigned to the teammates as detailed below. 7

1.5 Initial Project Management Roles 7

1.5.1 Team lead 7

1.5.2 Meeting Lead 7

1.5.3 Client Communications 8

2 Requirements 8

2.1 Problem Statement 8

2.1.1 Functional Requirements 8

2.1.1.1 Input each major's core classes per semester (a CPRE Freshman takes Math 165, Chem 167, CPRE 185, etc). Provide fields for course title, time requirement (50 minutes, 90 minutes, etc), and available spots 8

2.1.1.2 Add the remaining ECpE courses that are not required. 8

2.1.1.3 Create a scheduling algorithm that takes into account various constraints and preferences. Algorithm should be used to find a free period for a new course Avoid scheduling two courses with overlapping content at the same time. 8

Ensure that specific courses (as defined in the rules) are not scheduled back to back 8

2.1.1.4 Input the available times for the courses in the system. If possible, pull data from the classes.iastate.edu server. 8

2.1.1.5 Create a new course that needs to be scheduled. 8

2.1.1.6 When creating the new course, add "rules" that the class needs to find free periods around. Ex: don't conflict with other 6,7,8 semester CPRE core classes Ex: don't schedule the same time as another determined class 8

2.1.1.7 Allow for course time import from classes.iastate.edu (if possible) 8

2.1.1.8 Handle cross-listed courses correctly (it is the same course so don't treat it as two separate courses) 8

2.1.1.9 User shall be able to view current schedule 8

2.1.1.10 User shall be able to view class details 8

2.1.2 Resource Requirements 9

2.1.2.1 Modern Development Environment (IntelliJ IDE, Gitlab) 9

2.1.2.2 Library to develop a desktop application smoothly. 9

2.1.2.3 Program should be easy to install since it is going to be non-engineers installing the program (resource constraint) 9

2.1.3 UI Requirements and Qualitative Aesthetic Requirements 9

2.1.3.1 Desktop application 9

2.1.3.2 Easily read grid of already scheduled classes. 9

2.1.3.3 Menus to create classes and look for available times. 9

2.1.3.4 Ensure that schedules are formatted in a user-friendly and legible manner 9

2.3 Intended Users and Uses 9

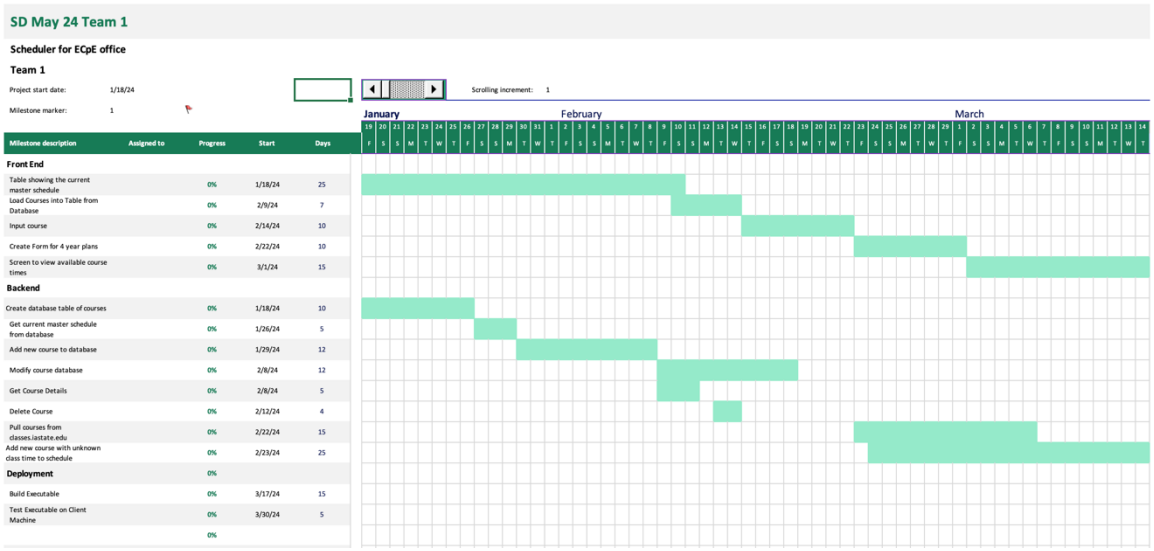
3 Project Plan 9

3.1 Task Decomposition 9

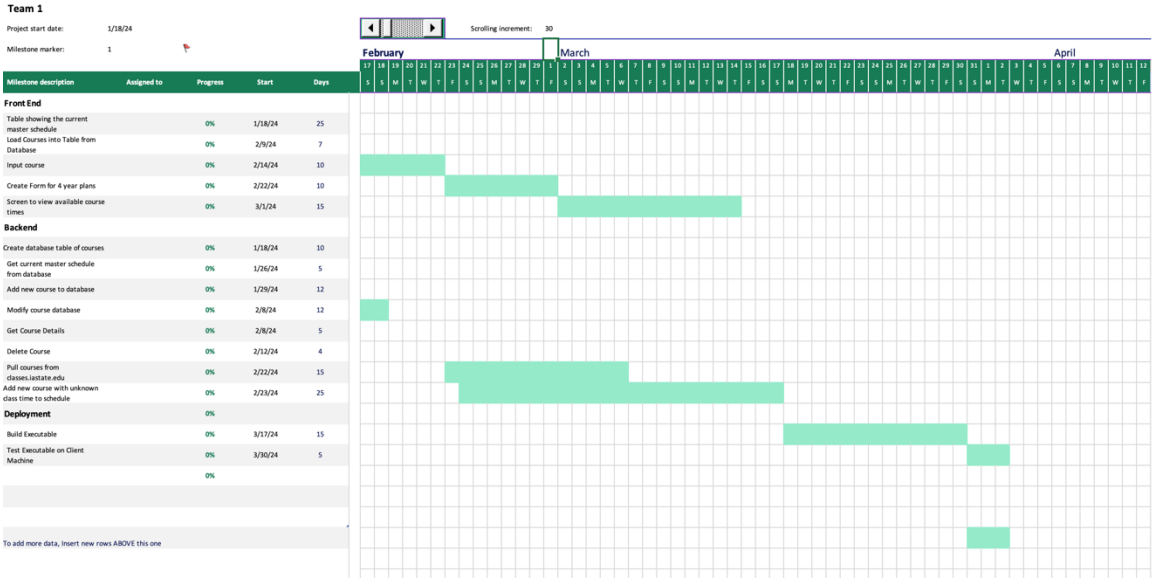
3.2 Project Management/Tracking Procedures 11

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 12

3.4 Project Timeline/Schedule 13



13



13

3.5 Risks And Risk Management/Mitigation 13

3.6 Personnel Effort Requirements 17

3.7 Other Resource Requirements 17

4 Design 17

4.1 Design Content 17

4.2 Design Complexity 18

4.3 Modern Engineering Tools 18

4.4 Design Context 19

4.5 Prior Work/Solutions 20
4.6 Design Decisions 20
4.6 Proposed Design 21
4.7 Design Visual and Description 24
4.8 Technology Considerations 24
4.9 Design Analysis 24
4.10 Appendix A – Time Slots 25
4.11 Appendix B – Input Format 25
4.12 Appendix C – Output Format 26
4.13 Appendix D – Database Design Diagram 27
5 Testing 28
5.1 Unit Testing 28
5.2 Interface Testing 28
5.3 Integration Testing 28
5.4 System Testing 28
5.5 Regression Testing 29
5.6 Acceptance Testing 29
5.7 Security Testing 29
5.8 Results 29
6 Implementation 29
7 Professionalism 29
7.1 Areas of Responsibility 30
7.2 Project Specific Professional Responsibility Areas 30
7.3 Most Applicable Professional Responsibility Area 30

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1. Figure 1: Task Decomposition Diagram
2. Figure 2: Gantt Chart

3. Figure 3: Effort Chart
4. Figure 4: Prototype Design
5. Figure 5: Add new class popup
6. Figure 6: How many students can take each course
7. Figure 7: Conflict screen

1 Team

1.1 Team Members

Simeon Steward

Brian Schomer

Lewis Callaway

Isaiah Ortiola

Carter Everts

Michael Less

1.2 Required Skill Sets for Your Project

Front-End Programming (specifically desktop applications), Backend Programming (including algorithms for scheduling/conflict graphs), communication with client, time management.

1.3 Skill Sets covered by the Team

All team members have gone through the same set of core classes and have an ability to troubleshoot and do full-stack development. This means there will be overlap with the skills.

Frontend: Michael Less

Backend: Simeon Steward, Brian Schomer

Testing: Carter Everts

Component Designer: Isaiah Ortiola

Client Communication: Lewis Callaway

Time Management: Simeon, Lewis, and Brian

1.4 Project Management Style Adopted by the team

We will follow the agile system with roles assigned to the teammates as detailed below.

1.5 Initial Project Management Roles

1.5.1 Team lead

Simeon Steward

1.5.2 Meeting Lead

Brian Schomer

1.5.3 Client Communications

Lewis Callaway

2 Requirements

2.1 Problem Statement

When new courses need to be added, the ECpE department needs to schedule classes in a manner where courses are scheduled at a time that does not conflict for most students. The schedule is currently scheduled manually by utilizing a master spreadsheet and testing potential times by hand. Additionally, the reliance on a master spreadsheet makes it difficult to quickly adapt without more manual attempts at finding a free time. A user-friendly desktop application that can streamline the course scheduling process for the ECpE department will reduce scheduling conflicts for students by intelligently assigning class times based on predefined rules and constraints. There is currently no automated system in place to optimize the scheduling classes so our solution will be new.

Requirements & Constraints

2.1.1 Functional Requirements

2.1.1.1 Input each major's core classes per semester (a CPRE Freshman takes Math 165, Chem 167, CPRE 185, etc). Provide fields for course title, time requirement (50 minutes, 90 minutes, etc), and available spots

2.1.1.2 Add the remaining ECpE courses that are not required.

2.1.1.3 Create a scheduling algorithm that takes into account various constraints and preferences.
- Algorithm should be used to find a free period for a new course
- Avoid scheduling two courses with overlapping content at the same time
- Ensure that specific courses (as defined in the rules) are not scheduled back to back

2.1.1.4 Input the available times for the courses in the system. If possible, pull data from the classes.iastate.edu server.

2.1.1.5 Create a new course that needs to be scheduled.

2.1.1.6 When creating the new course, add "rules" that the class needs to find free periods around.
Ex: don't conflict with other 6,7,8 semester CPRE core classes
Ex: don't schedule the same time as another determined class

2.1.1.7 Allow for course time import from classes.iastate.edu (if possible)

2.1.1.8 Handle cross-listed courses correctly (it is the same course so don't treat it as two separate courses)

2.1.1.9 User shall be able to view current schedule

2.1.1.10 User shall be able to view class details

2.1.2 Resource Requirements

2.1.2.1 Modern Development Environment (IntelliJ IDE, Gitlab)

2.1.2.2 Library to develop a desktop application smoothly.

2.1.2.3 Program should be easy to install since it is going to be non-engineers installing the program (resource constraint)

2.1.3 UI Requirements and Qualitative Aesthetic Requirements

2.1.3.1 Desktop application

2.1.3.2 Easily read grid of already scheduled classes.

2.1.3.3 Menus to create classes and look for available times.

2.1.3.4 Ensure that schedules are formatted in a user-friendly and legible manner

2.2 Engineering Standards

There are not many engineering standards apply to our project since it is a purely software project and it does not have to abide by any data protection standards since it does not deal with any type of protected information.

We plan to use Java and Spring Boot and will try to align with the respective programming standards. For Spring Boot we will separate the layers when possible to lead to smaller class files that are more easily maintained.

We are also going to attempt to make bug free code, but since bugs are in all code we will have to fix them as they pop up. We will make use of the sprint board to plan out which bugs have the largest impact and work to fix them as they pop up, leading to a much more stable product.

2.3 Intended Users and Uses

The ECpE office staff (Vicky and Tina in particular) are the primary users of this program. They will use this application to record the already scheduled courses from ECpE and enter in the core schedule for an ECpE student. There would be one for EE, one for CPRE, and one for CyberE. Next, a new course would need to be scheduled. A new course would be created and then any conflicts that need to be avoided would be entered. The program would then return the potential times for the new course.

3 Project Plan

3.1 Task Decomposition

https://docs.google.com/drawings/d/1oRg6CpgnTUuXv8lNf3QCAjmepitAHT3fR_11pdMPUHM/edit?usp=sharing

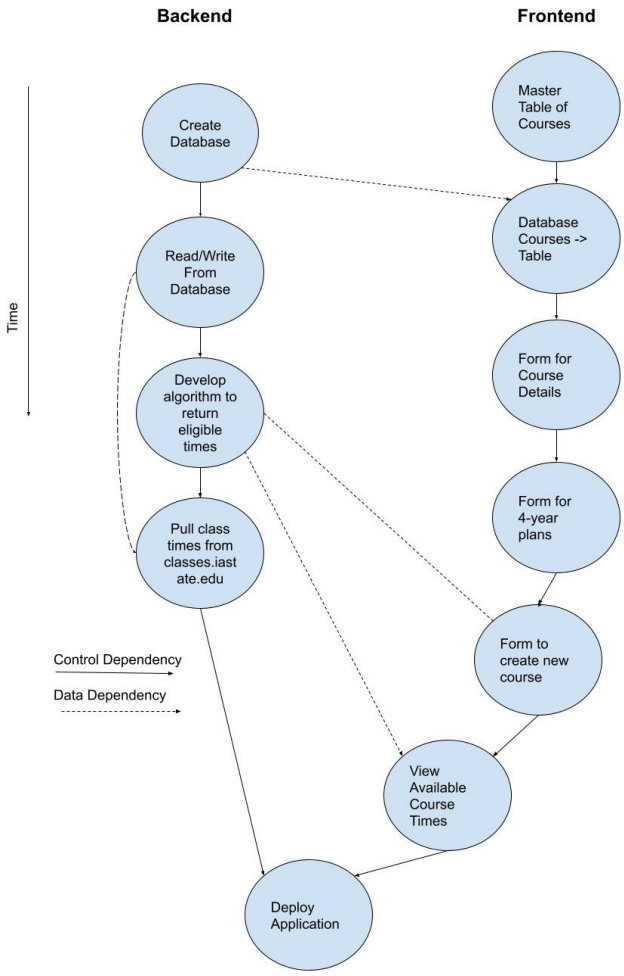


Figure 1: Task Decomposition Diagram

3.1.1 Front End - UI

3.1.1.1 Table showing the current master schedule for a specific semester (Req 3.2)

3.1.1.2 Make the table showing current master schedule read from backend (Req 3.2)

3.1.1.3 Form to input course, course times, and courses or schedules it can't conflict with (Req 1.6)

3.1.1.4 Form to input 4-year plan courses (Req 1.1)

3.1.1.5 Form to create a new course (Req 1.5)

3.1.1.6 Screen to view available course times that a new course could fit into (Req 3.4)

3.1.2 Front End - Backend Interface:

3.1.2.1 GET current master Schedule (Req 1.9)

3.1.2.2 POST new course with known class time (including class times, courses it can't conflict with) (Req 1.1, 1.2, 1.8)
3.1.2.3 PUT modify Course (including class times, courses it can't conflict with) (Req 1.1, 1.2, 1.8)
3.1.2.4 GET course details (including class times, courses it can't conflict with) (Req 1.10)
3.1.2.5 DELETE Course and associated data (Req 1.1, 1.2, 1.8)
3.1.2.6 [Extension] POST Add new course with unknown class time to current schedule (Req 1.5)
3.1.2.7 [Extension] GET possible schedules generated based on requirements (Req 1.3)
3.1.2.8 [Extension] POST Set current possible schedule to current master schedule (Req 1.3)

3.1.3 Backend - Business Logic

3.1.3.1 Get current master Schedule (Req 1.9)
3.1.3.2 Add new course (including class times, courses it can't conflict with) (Req 1.1, 1.2, 1.8)
3.1.3.3 Modify Course (including class times, courses it can't conflict with) (Req 1.1, 1.2, 1.8)
3.1.3.4 Get course details (including class times, courses it can't conflict with) (Req 1.10)
3.1.3.5 Delete Course and associated data (Req 1.1, 1.2, 1.8)
3.1.3.6 [Extension] Add new course with unknown class time to current schedule (Req 1.5)
3.1.3.7 [Extension] Generate possible schedules generated based on existing schedule (Req 1.3)
3.1.3.8 [Extension] Post Set current possible schedule to current master schedule (Req 1.3

3.1.4 Data Access Layer / ORM & Database

3.1.4.1 Create a database table of courses
    Course Name
    Degree Requirements i.e. SE semester 3
    Time & Duration
    Professor?
3.1.4.2 Develop a solution to pull from classes.iastate.edu (Req 1.7)
3.1.4.3 Deployment:
3.1.4.4 Create an executable that can be run on the client's computer (Req 3.1)

3.2 Project Management/Tracking Procedures

We are using waterfall+agile for this project. The reason it is waterfall+agile is that we do have a base level of code that needs to be written before it can even be confirmed with the client. We have clearly defined requirements upfront so this will give us a solid basis to build the software upon. We will check in with the client more frequently as soon as we have more visible changes and when it comes time to confirm changes were successful. Additionally, the general rule of thumb is that below a 200 person-day effort project, agile overhead is too much. So, we'll have a waterfall task decomposition and schedule but finish each task in an agile manner.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, Github, Trello, Slack or any other tools helpful in project management.

Our team will use GitLab to formally track progress as well as use Discord for more informal updates.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The milestones will be split into front and backend to try and keep as little dependency as possible

Front end (these do not have metrics as they are more visual tasks):

Display the current master schedule

Load the table to display from a database

Import new courses

Create 4 Year Plans

Screens to view available course times

Rudimentary Desktop interface

Make sure Input is taken

Ensure some sort of Output

Backend:

Create database table of courses – Needs to be a file that can loaded and transferred from one person to the next

Get current master schedule from database – After selecting the file, we do not want the set up/loading to take more than 10 seconds

Add new course to database – App should respond in .5 seconds

Modify course database – App should respond in .5 seconds

Get Course details – App should pull up new information in 1 second

Pull courses from classes.iastate.edu – Make minimal API calls, process returned information in 5 seconds

Add new course with unknown class time – We want the algorithm to find new spots within 10 seconds *NOTE: Start Development on the Algorithm early to ensure time to deal with any difficulties later on*

3.4 Project Timeline/Schedule

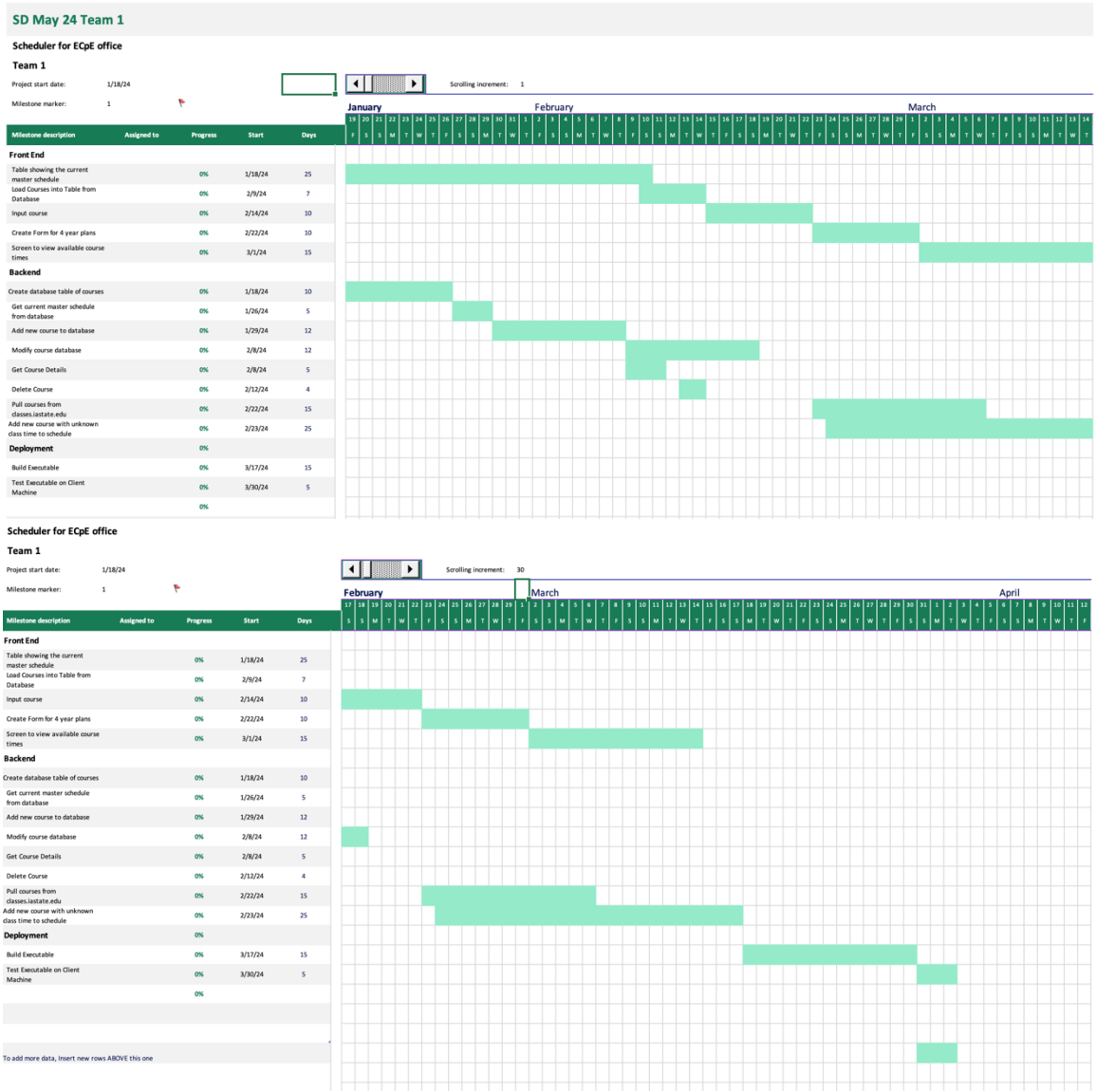


Figure 2: Gantt Chart

3.5 Risks And Risk Management/Mitigation

Consider for each task what risks exist (certain performance target may not be met; certain tool may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

Agile projects can associate risks and risk mitigation with each sprint.

Front End - UI

1. Table showing the current master schedule for a specific semester (Req 3.2)
   Risk: Data representation is unclear or formatted incorrectly.
   Probability: 0.4

2. Make the table showing current master schedule read from backend (Req 3.2)
   Risk: Backend connectivity issues or data mismatch.
   Probability: 0.5
   Mitigation Plan: Test connectivity rigorously. Consider adding a system for manual creation

3. Form to input course, course times, and courses or schedules it can't conflict with (Req 1.6)
   Risk: Users input incorrect data leading to scheduling conflicts.
   Probability: 0.6
   Mitigation Plan: Add input validation and error messages to guide users. Consider adding a review stage before data submission.

4. Form to input 4-year plan courses (Req 1.1)
   Risk: Mismatch between expected course sequences and actual university offerings.
   Probability: 0.4

5. Form to create a new course (Req 1.5)
   Risk: Duplication of courses due to different course codes but similar content.
   Probability: 0.5
   Mitigation Plan: Implement a course search functionality to check for existing courses before creation.

6. Screen to view available course times that a new course could fit into (Req 3.4)
   Risk: The algorithm doesn't find the optimal time slot, leading to inefficiencies.
   Probability: 0.4

Front End - Backend Interface

1. GET current master Schedule (Req 1.9)
   Risk: Incomplete or corrupted data transfer.
   Probability: 0.4

2. POST new course with known class time (including class times, courses it can't conflict with) (Req 1.1, 1.2, 1.8)

Risk: Data mismatch or incorrect storage due to invalid input or server error.
Probability: 0.4

3. PUT modify Course (including class times, courses it can't conflict with) (Req 1.1, 1.2, 1.8)
   Risk: Overwriting existing data unintentionally.
   Probability: 0.5
   Mitigation Plan: Always create a backup of existing data before modification. Version control for course details.

4. GET course details (including class times, courses it can't conflict with) (Req 1.10)
   Risk: Data might not reflect the most recent updates or could be incomplete.
   Probability: 0.4

5. DELETE Course and associated data (Req 1.1, 1.2, 1.8)
   Risk: Accidental deletion or corruption of associated data.
   Probability: 0.6
   Mitigation Plan: Add a validation screen before deletion and hold backups for undo's.

6. [Extension] POST Add new course with unknown class time to current schedule (Req 1.5)
   Risk: Conflicting data entries when course time becomes known.
   Probability: 0.5
   Mitigation Plan: Continually check to see if the now known times conflict.

7. [Extension] GET possible schedules generated based on requirements (Req 1.3)
   Risk: The generated schedules might not be optimal or could miss out on potential configurations.
   Probability: 0.5
   Mitigation Plan: Perform routine tests and gather user feedback to refine scheduling algorithms over time.

8. [Extension] POST Set current possible schedule to current master schedule (Req 1.3)
   Risk: Overwriting existing master schedule without proper validation.
   Probability: 0.6
   Mitigation Plan: Provide a preview before overwriting and maintain version history of schedules to roll back if needed.

Data Access Layer & Database

1. Create a database table of courses
   Risk: Data structure constraints might lead to issues with scalability or extensibility.

Probability: 0.4

2. Develop a solution to pull from classes.iastate.edu (Req 1.7)
   Risk: External website structure or data format changes, leading to pulling failures.
   Probability: 0.4

3. Create an executable that can be run on the client's computer (Req 3.1)
   Risk: Compatibility or performance issues on varying client systems.
   Probability: 0.3

3.6 Personnel Effort Requirements

		Hours Worked					
Tasks	**Est. Hours**	**Brian Schomer**	**Simeon Steward**	**Lewis Callaway**	**Carter Evans**	**Michael Less**	**Isaiah Ortiola**
Front End							
Table showing the current master schedule	20						
Load Courses into Table from Database	15						
Input course	15						
Create Form for 4 year plans	10						
Screen to view available course times	5						
Backend							
Create database table of courses	5						
Get current master schedule from database	15						
Add new course to database	15						
Modify course database	15						
Get Course Details	15						
Delete Course	10						
Pull courses from classes.iastate.edu	15						
Add new course with unknown class time to schedule	15						
Deployment							
Build Executable	5						
Test Executable on Client Machine	15						
	190						

Figure 3: Effort Chart

The table above shows the estimated hours for each task (as seen in the Gantt Chart) and the rightmost portion of the table will help our group members log the time they've put into each task.

3.7 Other Resource Requirements

Since our project is entirely a software project, there should be no need for any types of additional hardware or parts needed for our project. We shouldn't need any other physical materials to create our project, however one thing that we might require is some kind of server to allow for the application to be accessed at any time. This is only if we are unable to create a desktop application which is unlikely. Besides a possible server, there should be no other requirements needed for our project.

4 Design

4.1 Design Content

This section covers the design for the Class Scheduler's user interface and backend. The design will give an overview of the different parts of the overall software executable and describes the

components that will comprise the program. It also describes the broader context and design decisions that were made.

4.2 Design Complexity

Provide evidence that your project is of sufficient technical complexity. Use the following metric or argue for one of your own. Justify your statements (e.g., list the components/subsystems and describe the applicable scientific, mathematical, or engineering principles)

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles

> A. User Interface (UI): The development of a user-friendly desktop application requires expertise in user experience (UX) design, including layout design, accessibility, and information presentation principles.
> B. Data Management: Managing class schedules, course rules, and student data necessitates robust data handling principles, including database design and optimization.
> C. Scheduling Algorithm: The core of the application is the scheduling algorithm, which must incorporate mathematical and computational principles to find optimal class times, considering various constraints and preferences. This may include graph theory, constraint satisfaction, and optimization techniques.
> D. Integration with External Data Sources: If we can pull data from external servers like classes.iastate.edu, integrating and synchronizing this data requires knowledge of web APIs and data parsing.
> E. Software Framework: Using Electron.js as the framework adds another layer of complexity. Understanding the framework's architecture, APIs, and best practices is crucial for a successful implementation.

2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

> A. Current Process: We aim to replace a manual class scheduling process with an automated system. This will help with making the process faster and easier for new office staff.
> B. End User: Intending to make the application easy to install for non-technical users adds another layer of complexity.
> C. Scheduling Algorithm: The need to create a new course with customizable rules for scheduling, including avoiding conflicts and adhering to specific semester schedules, requires additional considerations that a generic scheduling algorithm may not consider.

4.3 Modern Engineering Tools

WebStorm IDE: WebStorm IDE is a powerful integrated development environment that will be used for coding and development. Its role included code editing and debugging.

Gitlab: Gitlab will be used for version control and continuous integration. It will play a pivotal role in managing the project's source code. We will collaborate, track changes, and merge code seamlessly using Gitlab.

.js: Electron.js has been chosen as the framework for building the desktop application. Its role was to enable the development of desktop applications using web technologies (HTML, CSS, and JavaScript). It also has the functionality to work independently of any servers which is a requirement for our desktop-based program.

DBMS: SQLite will be used as a database platform to store class information, preferences, and rules. It will interface with Electron.js using an API.

4.4 Design Context

Describe the broader context in which your design problem is situated. What communities are you designing for? What communities are affected by your design? What societal needs does your project address?

Our problem is solving a very specific need for the ECpE office, however if successful, it could easily be extended to other departments across Iowa State. Our project is addressing a need to assist in making the scheduling of classes easier for ECpE office staff.

List relevant considerations related to your project in each of the following areas:

Area	**Description**	**Examples**
Public health, safety, and welfare	While our project doesn't necessarily affect public health or safety, it does work to make an existing process less stressful for ECpE staff.	The ECpE office staff is able to schedule new courses quicker and easier than before Professors can know a potential time for a new course quicker since the office staff can find an available time quicker than before
Global, cultural, and social	The software does not offer many opportunities to reflect the values, practices, and aims of the cultural groups it affects, but will attempt to be aligned to the practices and aims of the ECpE department whenever possible.	Not super applicable to our project, but we will ensure that the software aligns with the current standards and processes for scheduling classes so as many students have access to courses as possible.
Environmental	The project is purely project, so there is negligible environmental impact however it may provide a small positive impact.	Since this is a purely software project, it will have negligible impact. It may save a small amount of paper if the department printed the existing schedule out to find available times manually.

Economic	Our project will provide an economic benefit because it will help streamline office processes and save time for the ECpE office staff.	This project will likely save the ECpE office staff time in scheduling classes that they can rededicate toward assisting students which in turn could help Iowa State retain more students. The project has no recurring costs since it is locally run.

4.5 Prior Work/Solutions

Iowa State has a "schedule planner" tool at classes.iastate.edu and AccessPlus has similar functionality. However, these tools are aimed at a student being able to create a schedule that fits within their requirements. These tools wouldn't work for our application. One reason is the workflow for planning a course. The aforementioned tools both help students schedule classes once they are entered in the master schedule. Our project helps the ECpE determine where new courses should be scheduled in the master schedule. Our project is not intended to generate student schedules. Additionally, our tool must consider different rules than the aforementioned schedulers since we are not attempting to create a schedule for a student, but rather schedule a course. For example, a prerequisite would be enforced on Access Plus, but our project can schedule a new course at the same time as a prerequisite.

4.6 Design Decisions

No backend server: We want to make the project as maintainable in the future as possible. Not requiring a server to be operational for the code to run would assist with that. The client requested a desktop application. Additionally, only one person will be using this software so there is no need for concurrent access. The project may pull data from the classes.iastate.edu API, but it will also be designed in a manner that can allow for manual class entry if the API changes or is unavailable. All the data that needs to persist in the project can be saved as a local file and doesn't need to be synced on the cloud.

Choosing Electron: Our group has selected Electron.js as the framework to build a desktop application. We selected this platform for multiple reasons. One reason is that it is cross-platform so it can be used on both Mac and PC which is a convenient feature since the University uses both platforms. It also allows us to create a user interface in a web framework such as React, that most of our team members have some familiarity with unlike a Desktop UI framework like JavaFX. For a backend of NodeJS, our team members a little experience, however it is simpler to learn and frequently used in industry. Electron does have slightly worse performance than a native application since it runs on top of a variant of Chromium, however our program will not use many resources so this should not be a problem.

Nodejs & Typescript Backend: For the backend, we choose to go with NodeJS and Typescript as it is commonly and successfully used with electron, and we are confident that it will suit our needs as a backend framework. Typescript is a familiar language, and for a project of this size the types are appreciated over vanilla JavaScript.

SQLite database: Our group selected SQLite as a database for our project. We picked this instead of using a static file like CSV since users will be much less likely to attempt to open the file and corrupt it. There are many SQLite libraries (potentially an ORM) that would allow us to interface with our backend code which will allow for more streamlined development. We chose SQLite over a more extensive database like MySQL is because we don't require as much overhead to install the database. To run MySQL, the end user would need to be running a MySQL server which would complicate the installation. For SQLite, we can have a single file that our program interacts with. SQLite doesn't allow for simultaneous users to connect to the database, but that is not needed for our application.

4.6 Proposed Design

Discuss what you have done so far – what have you tried/implemented/tested?

4.7. 1 Design 0 (Initial Design)

Design Visual and Description

Include a visual depiction of your current design. Different visual types may be relevant to different types of projects. You may include: a block diagram of individual components or subsystems and their interconnections, a circuit diagram, a sketch of physical components and their operation, etc.

Describe your current design, referencing the visual. This design description should be in sufficient detail that another team of engineers can look through it and implement it.

Justify each component in the design with respect to requirements.

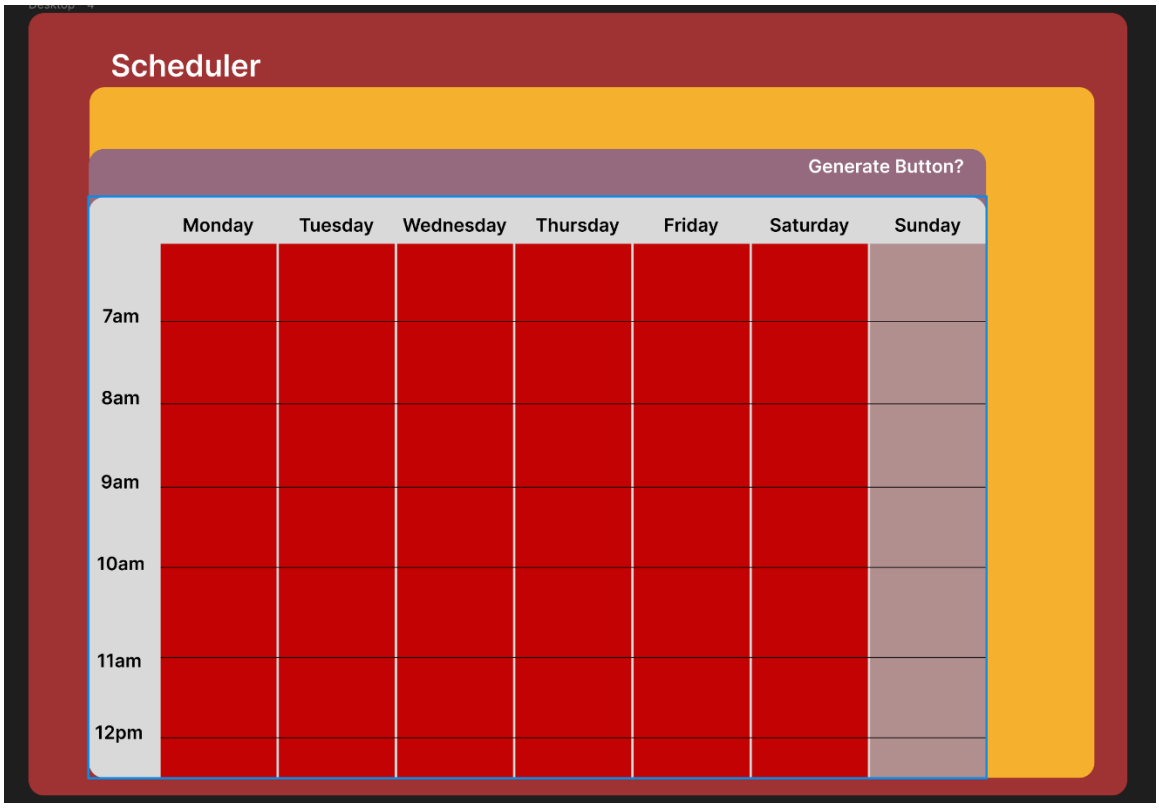


Figure 4: Prototype Design

This design is a rough idea for how our desktop app may look. Since this is Design 0, this is a very rough idea of what it will look like. This design includes a clean UI that is easy on the eyes and includes the ISU colors since it will be used by the ECpE department here on campus. The UI includes a clickable weekly schedule and possible button to randomly generate a schedule for the user.

Functionality

Describe how your design is intended to operate in its user and/or real-world context. This description can be supplemented by a visual, such as a timeline, storyboard, or sketch. How well does the current design satisfy functional and non-functional requirements?

This design is intended to let the user insert the desired courses into the weekly schedule where they choose. However, there is also a button that the user can use to generate a schedule which would be far less time consuming and would automatically generate a schedule with the desired classes that the user can then use for their scheduled courses for the semester. We believe that this design does a decent job satisfying the user's needs but could be improved.

4.7.2 Design 1 (Design Iteration)

Include another most matured design iteration details. Describe what led to this iteration and what are the major changes that were needed in Design 0.

New Class Setup:

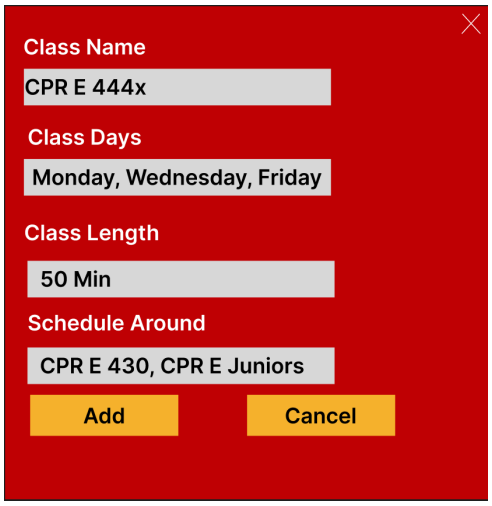


Figure 5: Add new class popup

Fitness of the new course:

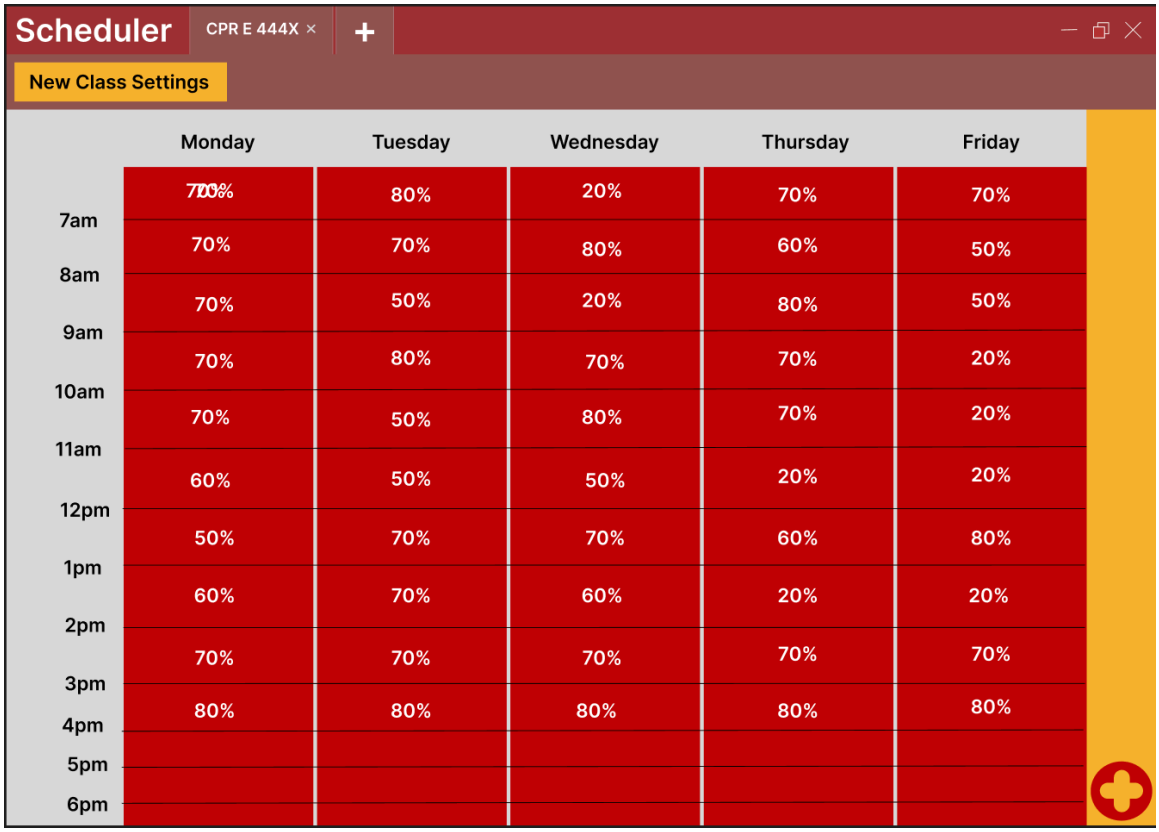


Figure 6: How many students can take each course

Clicking on a timeslot shows the potential conflicts.

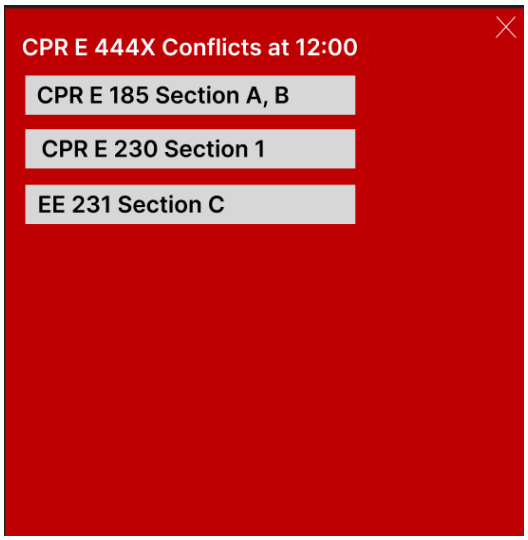


Figure 7: Conflict screen

4.7 Design Visual and Description

Include a visual depiction of this design as well highlighting changes from Design 0. Describe these changes in detail. Justify them with respect to requirements.

Some changes we made from Design 0 is that we removed the weekend slots since classes will not be scheduled on the weekend. We also changed the time that classes can be scheduled at since classes start as early as 7:45 am, and end as late as 6:20 pm. Another thing that we added was a way to test multiple classes at once with the addition of the '+' button. We created an input box that is used to create a new course that will be evaluated in the schedule. Then the output will provide the fitness of each time slot for a new course. Finally, clicking on a time slot will tell you the class sections that would be a conflict.

NOTE: The following sections will be included in your final design document but do not need to be completed for the current assignment. They are included for your reference. If you have ideas for these sections, they can also be discussed with your TA and/or faculty adviser.

4.8 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

4.9 Design Analysis

- Did your proposed design from 4.7 work? Why or why not?
- What are your observations, thoughts, and ideas to modify or iterate further over the design?

4.10 Appendix A – Time Slots

50 Minute Classes	75-Minute Classes
7:45am - 8:35am	8:00am - 9:15am
8:50am - 9:40am	9:30am - 10:45am
9:55am - 10:45am	11:00am - 12:15pm
11:00am - 11:50am	12:40pm - 1:55pm
12:05pm - 12:55pm	2:10pm - 3:25pm
1:10pm - 2:00pm	3:40pm - 4:45pm
2:15pm - 3:05pm	4:10pm - 5:25pm
3:20pm - 4:10pm	
4:25pm - 5:15pm	
5:30pm - 6:20pm	

4.11 Appendix B – Input Format

Two csv files are used for each major (Electrical, Computer, Cyber).

The first includes the class list with the columns Course ID, Section ID, Weekdays, Meeting Times, student capacity, and Associated Sections. This file can be shared among the three majors. If a connection to classes.iastate.edu is unable to be made, then ECpE staff will have to enter this information manually. Our team will start with manual entry to begin with. The ISU transition to WorkDay may standardize this process so there will be more clarification once a response from the WorkCyte team is received. If courses are scheduled like COM S 311, then the lecture and recitation are separate. In this case, we need to link each lecture section with the potential recitation/lab sections. Courses could also be like CHEM 167 where the recitation and lecture are combined into one section. According to the client, the courses will all be like COM S 311 with the WorkDay transition, but confirmation is being obtained.

Course ID	Section ID	Week Days	Meeting Times	Associated Section(s)	
CHEM 167	1	MWF, T	9:55 AM - 10:45 AM, 8:00 AM - 8:50 AM		
CHEM 167	3	MWF, T	9:55 AM - 10:45 AM, 9:00 AM - 9:50 AM		
CPRE 281	1	MWF, T	2:15 PM - 3:05 PM, 8:00 AM - 10:50 AM		
COM S 311	A	TTH	8:00 AM - 9:15 AM	1,2,3,4	
COM S 311	1	W	7:45 AM - 8:35 AM		

The second file is the list of classes for each of the three majors that are on the “traditional” flowchart. Each column has the classes listed that are traditionally taken for each of the 8 semesters. General education and technical electives are excluded from this list since students take very different courses for these and if a new course wants to avoid the time of another technical elective, the course should be entered into the program as an individual conflict.

SEM1	SEM2	SEM3	SEM4	SEM5	SEM6	SEM7	SEM8
CHEM 167	COM S 227	CPR E 281	CPR E 288	CPR E 232	CPR E 308	CPR E 491	CPR E 492
CPR E 185	CPR E 166	COM S 228	E E 230	CPR E 381	ENGL 314	STAT 330	
ENGL 150	MATH 166	E E 201	CPR E 310	COM S 311	COM S 309		
ENGR 101	ENGL 250	MATH 267	MATH 207		CPR E 494		
LIB 160	PHYS 231						
MATH 165	PHYS 231L						

Additional inputs on the GUI would be:

1. How long is the new course (60 min or 75 min)?
2. What is the capacity of the new course?
3. Target major (ECpE, EE, Cyber E) and year(s), any individual classes to avoid scheduling around

4.12 Appendix C – Output Format

The output of this program will show a list of recommended time slots, each with their fitness score and a breakdown of the conflicts that it has.

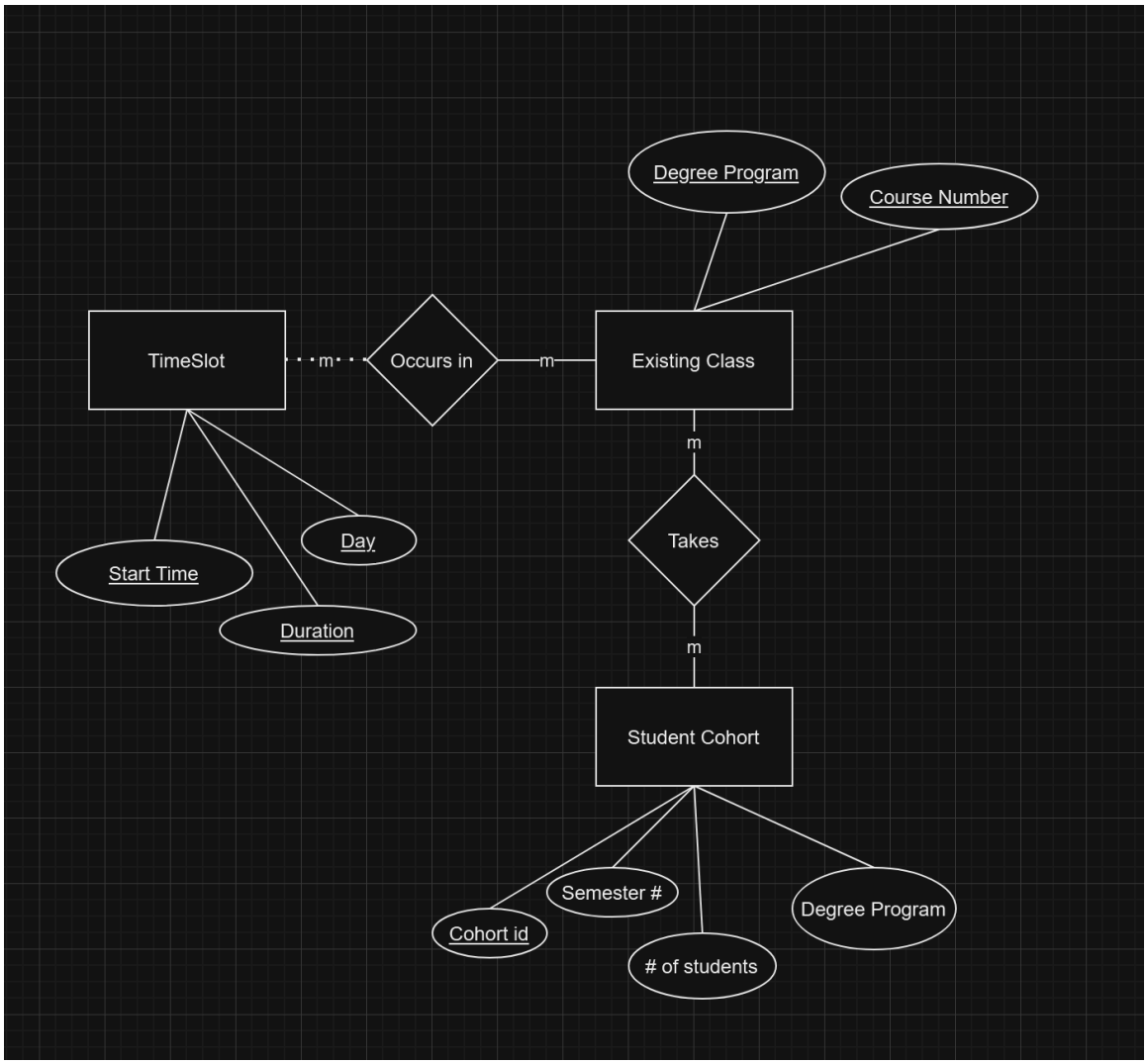
The fitness score will be based around the schedule availability of target major and grade level (ex. CPRE Junior 1st Semester) and any other individual conflicts that exist.

Pseudo Code:

```
class potentialTimeSlot {
   course:{
      name,
      department,
      coursename
   },
  Timeslot,
  Fitness score,
  reasonList<reasons>{
        "Conflicts with 25% of class x which that CPRE juniors will take"
  }
}
```

Best Fit Examples	
50 Minute Classes	% Juniors Eligible
7:45am - 8:35am	84
8:50am - 9:40am	59
9:55am - 10:45am	71
11:00am - 11:50am	75
12:05pm - 12:55pm	72
1:10pm - 2:00pm	77
2:15pm - 3:05pm	68
3:20pm - 4:10pm	59
4:25pm - 5:15pm	75
5:30pm - 6:20pm	55

4.13 Appendix D – Database Design Diagram



5 Testing

5.1 Unit Testing

Unit testing will be used to test critical functions within our code. For functions that could easily be messed up, a unit test will be put into place to ensure that other code is not unintentionally broken. While we don't know every specific function that will be in place, we have identified some known functions this will be good for:

- The scheduling algorithm functions – ensure parts of the scheduling algorithm produces the desired input and output.
- Test the input of core classes for different majors to ensure that the course title, time requirement, and available spots are correctly stored in the database (Req 2.2.1.1)
- Verify that the application handles different time formats correctly (Req 2.2.1.1)
- Test the creation of a new course to ensure that it is correctly added to the database (Req 2.2.1.2)
- Verify a rule is correctly enforced during scheduling (Req. 2.2.1.6)
- Test the handling of cross listed courses to ensure they are not treated separately (Req. 2.2.1.8)

For a tool, we will likely use Jest as it is the most popular unit testing platform for JavaScript programs. Hopefully our team can enable it to run on a CI/CD pipeline.

5.2 Interface Testing

Our program is not a client and server communication-based application, so there is not a need to ensure that the API is matching a standard protocol. The only API we could check would be ensuring that classes.iastate.edu is accessible and matches the previously utilized data standard.

5.3 Integration Testing

A potential integration test for our project would be ensuring that a standardized CSV format is read properly and saved to the database. This would ensure that the data is not harmed in the process and verify it can be accurately retrieved. Another integration test could be the integration between the user interface and the core application logic. Verify that user actions in the UI (e.g., creating a new course, setting rules) correctly trigger the corresponding functions in the backend.

We plan to use Selenium and Jest in order to test that the changes are accurately reflected.

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?'

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

We plan on creating groups of multiple unit tests and then running them together to create a test to ensure full system functionality. There then can be integration tests that create sample user inputs and then test the output of the program to ensure that the desired output is produced.

The CI/CD pipeline on Gitlab will be used to ensure that the code is working as intended. Upon pushing code to github, all code will be tested to verify that if it passes the implemented tests. This

way we can confirm that code modifications doesn't negatively impact the results. The individual unit tests align to specific requirements so a group of smaller units or integration tests will work to ensure requirements are being met.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure do not break? Is it driven by requirements? Tools?

The unit tests will be run when code is being built to determine if changes have broken something else (related or unrelated). The requirements are being used to develop the tests, so a failure of a test will mean a violation of a requirement. We will use internal code interfaces to provide some separation of concerns which will help later in our development when we wish to add or change code but do not want to break our existing functionality. This also means that our job of mocking and testing will be easier later and our individual components will not affect each other.

5.6 Acceptance Testing

We plan to show the client working demonstrations in the development process. Even though specific parts of the code will not be functioning, we can determine if the specific parts of the project under test are aligning to what the client desires. Since our project is just used by a specific client, we can ensure it is exactly what they want.

5.7 Security Testing

Security testing will not be applicable for this project since we are not storing any private information. Because everything we will be working on for the class scheduler is already available to the public, we won't need to worry about any type of security testing.

5.8 Results

The expected results of testing would be that the app passes all of the tests that we write. There are no results at this point in time as development has not yet started for the app.

6 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3. If your project has inseparable activities between design and implementation, you can list them either in the Design section or this section.

7 Professionalism

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Pick one of IEEE, ACM, or SE code of ethics. Add a column to Table 1 from the paper corresponding to the society-specific code of ethics selected above. State how it addresses each of the areas of seven professional responsibilities in the table. Briefly describe each entry added to the table in your own words. How does the IEEE, ACM, or SE code of ethics differ from the NSPE version for each area?

If we take the Software Engineering code of ethics and compare it to Table 1 from the paper discussed above, then we can see that many of the principles seen in the SE code of ethics are able to relate to the items seen in the table. For example the first item in the table is Work Competence which is defined as "Perform work of high quality, integrity, timeliness, and professional competence.", this can be closely compared to Principle 1: Product from the SE code of ethics, which summarily states that software engineers must assure that the product they work on is both useful and acceptable to the public, client, and/or user and should be completed on time and with little to no error.

7.2 Project Specific Professional Responsibility Areas

For each of the professional responsibility area in Table 1, discuss whether it applies in your project's professional context. Why yes or why not? How well is your team performing (High, Medium, Low, N/A) in each of the seven areas of professional responsibility, again in the context of your project. Justify.

The first item of the table Work Competence does apply to our profession context, especially because we are the second group to have touched this project, but our predecessors were not able to deliver a completed product, thus adding more pressure on us to deliver a quality software product to our client. Our team is performing quite well in being able to deliver quality work, so far only in reports, on a timely basis and it's expected to continue going forward in the second semester.

7.3 Most Applicable Professional Responsibility Area

The most applicable professional responsibility area for our team would be the SE code of ethics since our project is exclusively a software project.

7.4.1 Team Contract

Team Members:

1) Brian Schomer 2) Lewis Callaway

3) Simeon Steward 4) Isaiah Ortiola

5) Michael Less 6) Carter Everts

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-

mail, phone, app, face-to-face):

3. Decision-making policy (e.g., consensus, majority vote):

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

3. Expected level of communication with other team members:

4. Expected level of commitment to team decisions and tasks:

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

2. Strategies for supporting and guiding the work of all team members:

3. Strategies for recognizing the contributions of all team members:

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

2. Strategies for encouraging and support contributions and ideas from all team members:

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

2. Strategies for planning and assigning individual and team work:

3. Strategies for keeping on task:

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

2. What will your team do if the infractions continue?

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Brian Schomer DATE 12-3-23

2) Lewis Callaway DATE 12-3-23

3) Simeon Steward DATE 12-3-23

4) Isaiah Ortiola DATE 12-3-23

5) Michael Less DATE 12-3-23

6) Carter Everts DATE 12-3-23