myADVISE
System Requirements Specification

Version 1.0
December 7, 2016

1. Introduction

1.1. Purpose

The purpose of this document is to:

1) Explain the high-level and detailed requirements of the myADVISE web application

2) Outline myADVISE program flow and usage

This document is designed not only to assist those using the myADVISE software, but it also aims to aid developers in the design process. This document should serve as a preliminary roadmap for software design and should be modified as requirements are discovered/deleted.

1.2 Scope

The myADVISE software is a web application designed to assist students in the class scheduling process. The software will generate a class schedule for the user based on different conditions. The software thus aims to eliminate the unnecessary stress caused by scheduling and registration; it also serves as an aid to advisors and will help them suggest a schedule to students based on specific needs.

1.3 Definitions, Acronyms, and Abbreviations

UofL	University of Louisville
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol Secure
CRUD	Create, Read, Update, Delete
Flight Plan	UofL Graduation Plan for students

MVC	Model-View-Controller

1.4 References

1) IEEE Recommended Practices for Software Requirement Specification

1.5 Overview

The remainder of this document analyzes the myADVISE software in more detail. It shall examine topics including high-level and specific requirements, design, and overall functionality.

2. Overall description

2.1 Product Perspective

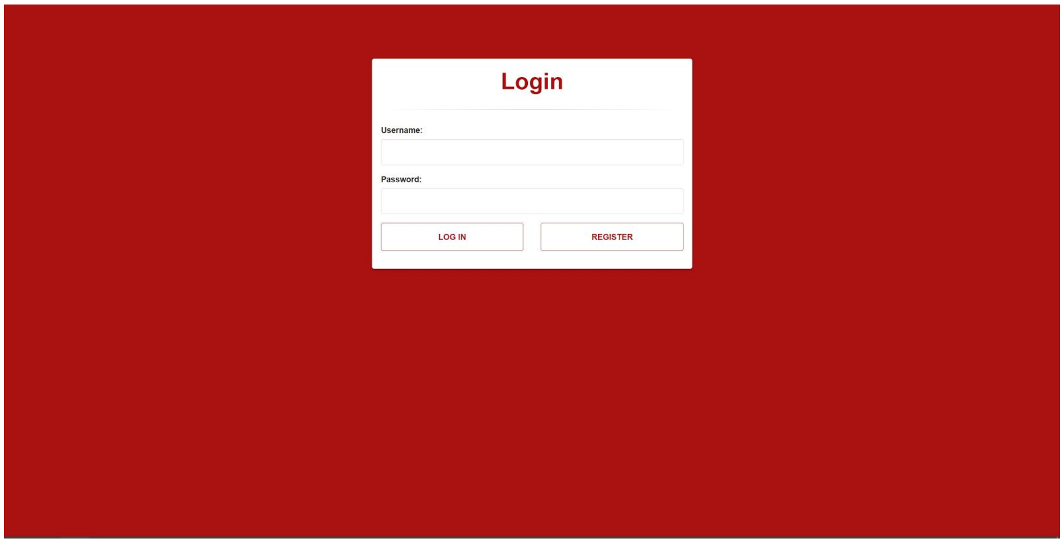
The myADVISE software is self-contained and independent. With that being said, myADVISE development interfaces with external software such as frameworks and databases. While these softwares are required for development and implementation, the end user need only a web browser and an internet connection to access myADVISE.

2.1.1 System Interfaces

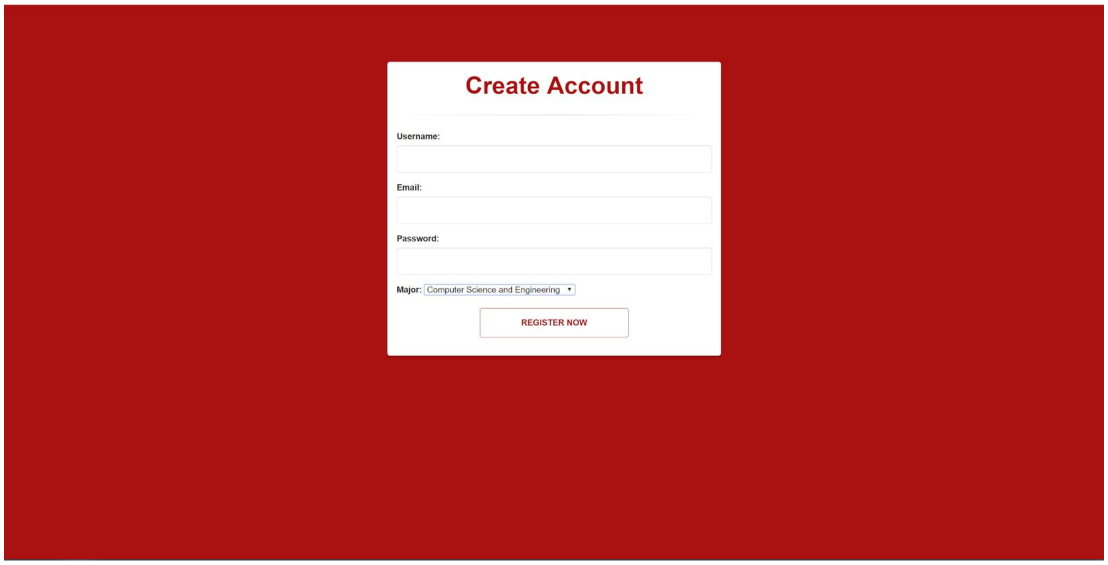
As explained in section 2.1, myADVISE is self-contained and independent, though it does require interfacing with external software. The application interfaces with a Heroku web host which is responsible for serving pages, handling external and internal requests, and housing the database. A Postgres database is used for data acquisition and storage purposes. Living on the Heroku server, the Postgres database is limited to 10,000 rows, and is modified through the use of a web framework detailed in subsequent sections.

2.1.2 User Interfaces

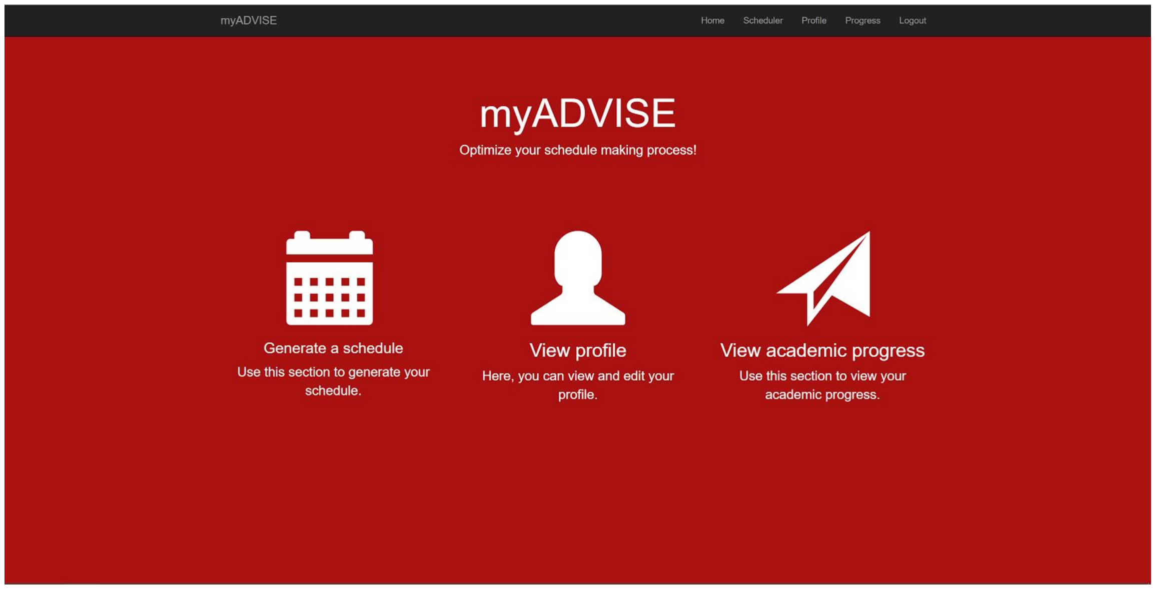
The user interface for the myADVISE application consists of 5 primary webpages. The login page is used for existing users to login and for new users to create accounts. A mockup of the login page is included below:



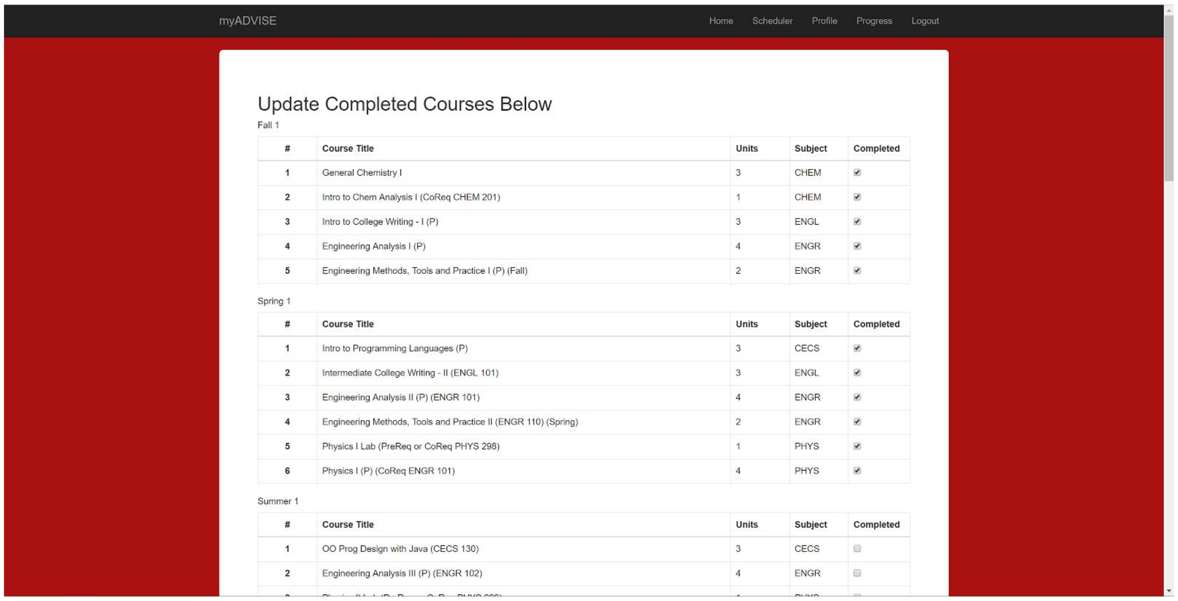
Webpage visitors that want to create accounts are directed to the registration page from the login page. The registration page allows users to enter a username, email, password and choose a University of Louisville Speed School of Engineering major. A mockup of the registration page is included below:



The homepage allows authenticated users to either A.) generate a schedule, B.) View their user profile, or C.) view their academic progress. A mockup of the home page is included below:


myADVISE
Home
Scheduler
Profile
Progress
Logout
myADVISE
Optimize your schedule making process!
Generate a schedule
Use this section to generate your schedule.
View profile
Here, you can view and edit your profile.
View academic progress
Use this section to view your academic progress.

The user progress page allows users to choose classes they have taken or not taken. The page displays a clickable row for each course in the appropriate UofL Flight Plan based on the user’s selected major. A mockup of the progress page is included below:



Update Completed Courses Below

Fall 1

#	Course Title	Units	Subject	Completed
1	General Chemistry I	3	CHEM	✓
2	Intro to Chem Analysis I (CoReq CHEM 201)	1	CHEM	✓
3	Intro to College Writing - I (P)	3	ENGL	✓
4	Engineering Analysis I (P)	4	ENGR	✓
5	Engineering Methods, Tools and Practice I (P) (Fall)	2	ENGR	✓

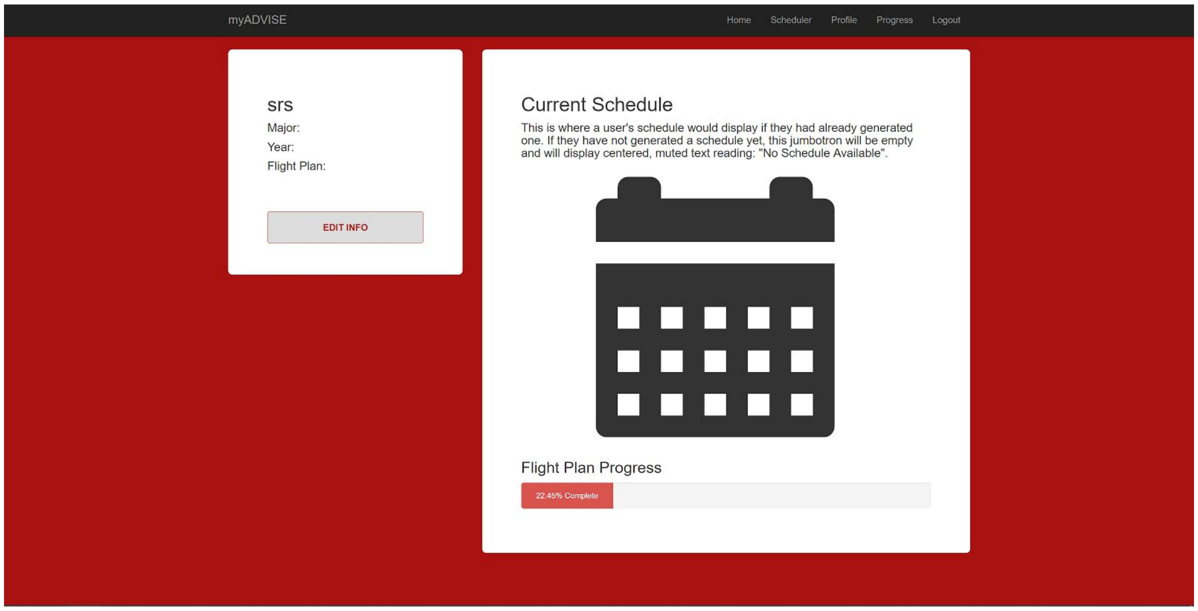
Spring 1

#	Course Title	Units	Subject	Completed
1	Intro to Programming Languages (P)	3	CECS	✓
2	Intermediate College Writing - II (ENGL 101)	3	ENGL	✓
3	Engineering Analysis II (P) (ENGR 101)	4	ENGR	✓
4	Engineering Methods, Tools and Practice II (ENGR 110) (Spring)	2	ENGR	✓
5	Physics I Lab (PreReq or CoReq PHYS 298)	1	PHYS	✓
6	Physics I (P) (CoReq ENGR 101)	4	PHYS	✓

Summer 1

#	Course Title	Units	Subject	Completed
1	OO Prog Design with Java (CECS 130)	3	CECS	☐
2	Engineering Analysis III (P) (ENGR 102)	4	ENGR	☐

The user profile page allows users to edit their major, graduation year, and flight plan. It displays the user's most recently generated schedule. It also displays a progress bar that indicates how close the user is to completing their UofL Flight Plan. The fill of the progress bar is based on the number of courses the user has completed.



srs

Major:

Year:

Flight Plan:

Current Schedule

This is where a user's schedule would display if they had already generated one. If they have not generated a schedule yet, this jumbotron will be empty and will display centered, muted text reading: "No Schedule Available".

Flight Plan Progress

2.1.3 Software Interfaces

1) Heroku Web Host

a) Mnemonic: Heroku

b) Version Number: N/A

c) Source: https://www.heroku.com/

d) Purpose: Heroku is a web hosting platform that will house the myADVISE application and server

2) Postgres

a) Mnemonic: PG

b) Version Number: 9.5

c) Source: https://www.postgresql.org

d) Purpose: Primary relational database management system

2.1.5 Communications Interfaces

Network communications are accomplished through REST. REST is a client-server architecture which (among other things) leverages the full capacity of the HTTP protocol. This protocol handles data management between client-server both concurrent and persisting.

- Each URL on the server represents a resource; either a collection resource or an element resource.
- Different HTTP methods are used for different CRUD operation
- a PUT is a write/modify operation
- a POST is a create operation:
- a GET is a read/new operation
- State is not stored on the server-side. State is in the representations passed back and forth by the client's requests and the server's responses.

2.1.6 Memory Constraints

The primary bottleneck of memory and storage for the myADVISE project will be the storage on the database. The free version of Postgres hosting allows up to a maximum of 10,000 rows of storage. The distributed RAM from the hosting service may also pose technical issues in unlikely cases. All of the hosted solutions are entirely scalable which allows memory and computing restraints to be solved easily. In general, however, the myADVISE application has no control over these constraints; memory is completely dependent on the Heroku server.

2.1.7 Operations

The user is required to create a user profile. The user is required to enter preferences for classes and instruct the web page to generate a class schedule. The user is required to update their own course history.

2.2 Product Functions

The product provides the user with:

1) Ability to generate class schedules based on the following:
    a) Preferences (time of day, etc.)
    b) Major Flight Plan
2) Ability to track degree progress; degree progress determined by course history and courses remaining for major
3) Visual representation of generated course schedule

2.3 User Characteristics

The myADVISE web application is intended for University of Louisville students and staff associated with the students' course scheduling and Flight Plan handling. Users will require basic knowledge of web browsing and the university Flight Plan for the student's intended major.

2.4 Constraints

1) Reliability: students will depend on application to provide them with accurate course schedules so that students will not be applied to overlapping classes and will graduate on schedule.
2) Memory: Currently only 10,000 rows of memory can be utilized in the current system. Scaling is subject to costs.
3) Time: this application requires that a certain set of application criteria are met on or before the deadline. Features and tools that are not mission critical will be susceptible to available resources. For example, a schedule must be returned to the user in a reasonable amount of time.
4) Server Dependencies: our application is heavily dependent on database access. Server uptime is a vital aspect of product viability. Regardless of the features and tools created, without having a reliable database providing course information these tools will be useless.
5) Course availability: UofL course database is not readily available to users; methods to overcome this limitation are required.

2.5 Assumptions and Dependencies

1) This application will be accessed by a future or current UofL Speed School students and staff.
2) The UofL scheduling systems will be parsable.
3) The user is going to access this application through web browsers supported by bootstrap.
4) The hosting service will give reliable uptime for database and server.
5) The user’s of this application will be English speaking.
6) The user will be pursuing a single major.
7) The user only requires a schedule for upcoming semester.
8) The user is on the most recent Flight Plan for their respective major.
9) Summer semesters are not included.

2.6 Apportioning of Requirements

Visualization of schedules may be delayed to future releases. Full user preference functionality may be delayed to future releases. Functionality for additional majors/Flight Plans will be delayed to future releases.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Users are directed to the login page when they visit the myADVISE website. Returning users can login on this page with their username and password. If the user is a first time user, they can access the registration page via the login page. The registration page allows the new user to enter a username, password, and an email; it also allows the new user to select their major. Once the new account is created, the user is directed to the homepage. Returning users are also directed to the homepage after they have logged in.

From the home page, the user has three options. The first option is to generate a schedule. If the generate schedule option is selected, the application will generate a schedule based on the user's current course history and preferences. The user will be directed to their profile page where they can view the newly generated schedule.

The second option available to the user is viewing their profile. On the user profile page, users can edit their basic information and preferences, which includes changing their major. Users can also view their most recently generated schedule. Additionally, users can view their degree progress in the form of a progress bar located on the profile page. Users are also directed to the profile page after generating a schedule.

Finally, the user can visit the progress page. The progress page allows users to select the courses they have currently taken. The courses provided are based on the user's selected flight plans. The user's information is updated with these courses if the changes are committed.

3.1.2 Software Interfaces

Django is a web framework that facilitates a significant amount of functionality at all levels of the myADVISE web application. It separates, organizes, generates, and authenticates front and backend code. Django organizes developmental layers into a typical MVC layout to provide a logical layout and promote modularity. Django directly communicates with the data source, generating queries and schemas based off of developer input. Django provides an expansive testing suite, allowing for easy test creation and management.

Twitter Bootstrap is a CSS library that provides an expansive set of tools and design elements that greatly increase a website's user experience. It empowers developers to create an immersive experience without extensive design expertise. Bootstrap lives as an extension of the application's view layer. Together with Django's HTML generation capabilities, Bootstrap is used to design a dynamically scaling frontend environment for the rapid and varied developmental needs of the myADVISE application.

3.1.4 Communication Interfaces

Communication between internal and external interfaces is handled by the Django framework. It handles both server requests as well as the communication between controllers, models, views, and the data source (database).

3.2 System Features

3.2.1 Create Account

3.2.1.1 Introduction

The application shall allow users to create accounts.

3.2.1.2 Stimulus/Response Sequence

While on the login page, the option to register shall be provided to the user. Upon clicking register, the user is provided with a form such that they can enter a username, password, email and major. Upon submitting this form, a new account is created.

3.2.1.3 Associated Funtional Requirements

3.2.1.3.1 Functional Requirement 1

The login page shall provide the user the ability to create a new account.

3.2.1.3.2 Functional Requirement 2

The registration page shall allow the user to enter a username, password, email, and major.

3.2.1.3.3 Functional Requirement 3

The registration page shall not allow the user to create an account with a pre-existing username.

3.2.1.3.4 Functional Requirement 4

The registration page shall ensure all relevant data is completed before creating a new account.

3.2.1.3.5 Functional Requirement 5

Upon submission of the form, a new user profile is added to the myADVISE database containing the user’s entered information.

3.2.2 Logon

3.2.2.1 Introduction

The login page shall allow returning users to login with username and password.

3.2.2.2 Stimulus/Response Sequence

The user that wishes to login enters their username and password. The Django framework authenticates the username/password combination with what is stored in the database. If authentication is successful, the user is redirected to the homepage. Otherwise, the user is prompted to try again.

3.2.2.3 Associated Functional Requirements

3.2.2.3.1 Functional Requirement 1

The login page shall provide the user an option to enter a username and password.

3.2.2.3.2 Functional Requirement 2

The Django framework shall be used to attempt authentication with the entered username/password combination with existing values in the database.

3.2.2.3.3 Functional Requirement 3

Upon successful authentication, the user shall be directed to the homepage.

3.2.2.3.4 Functional Requirement 4

Upon authentication failure, the user shall be prompted to try again.

3.2.3 User Progress

3.2.3.1 Introduction

The user can select which courses they have taken in accordance with their specific Flight Plan.

3.2.3.2 Stimulus/Response Sequence

The progress page shall generate clickable rows for each course in the appropriate Flight Plan for the user. The user can select/deselect courses. Upon commit, the JSON object representing the user’s course history is updated in the database with the new information.

3.2.3.3 Associated Functional Requirements

3.2.3.3.1 Functional Requirement 1

The progress page shall provide clickable rows for each course in the Flight Plan appropriate for the user’s major.

3.2.3.3.2 Functional Requirement 2

The user shall have the option to select/deselect each course.

3.2.3.3.3 Functional Requirement 3

The user shall be able to commit changes on the progress page.

3.2.3.3.4 Functional Requirement 4

Upon commit, the JSON object representing the user’s course history shall be updated in the database with the new information.

3.2.4 User Profile Page

3.2.4.1 Introduction

The application shall provide the user with a basic profile page.

3.2.4.2 Stimulus/Response Sequence

The profile page will provide the user with the option to update his/her basic data and user preferences. The user will also be able to view his/her most recently generated schedule. A progress bar displaying the user’s Flight Plan progress will also be displayed.

3.2.4.3 Associated Functional Requirements

3.2.4.3.1 Functional Requirement 1

The profile page shall provide the user the option to update basic data.

3.2.4.3.2 Functional Requirement 2

The profile page shall provide the user the option to update their schedule preferences.

3.2.4.3.3 Functional Requirement 3

Upon update of basic data or user preferences, the database shall be updated with the newly entered information.

3.2.4.3.4 Functional Requirement 4

If the user chooses to change their major, they will be warned that all course history entered will be lost.

3.2.4.3.5 Functional Requirement 5

Modifiable preferences available shall include desired class times and ideal course load.

3.2.4.3.6 Functional Requirement 6

If a schedule has been generated, the schedule shall be displayed.

3.2.4.3.7 Functional Requirement 8

A progress bar shall be displayed that indicates the user’s proximity to Flight Plan completion.

3.2.4.3.8 Functional Requirement 9

A progress bar shall be based on completed courses in the user’s stored course history.

3.2.5 Schedule Generation

3.2.4.1 Introduction

The application shall allow the user to generate a schedule for the upcoming semester. This generation is based on completed courses and user preferences.

3.2.4.2 Stimulus/Response Sequence

Schedule generation shall use the user’s stored course history and preferences to generate a schedule for the upcoming semester. Upon completion, the user is returned to their profile page where the newly generated schedule is displayed.

3.2.4.3 Associated Functional Requirements

3.2.4.3.1 Functional Requirement 1

The scheduler shall choose classes that move the user closer to Flight Plan completion.

3.2.4.3.2 Functional Requirement 2

The scheduler shall grant higher priority to classes that satisfy user preferences.

3.2.4.3.3 Functional Requirement 3

The scheduler shall attempt to generate a schedule that includes a number of credit hours equal to or less than the user’s specified ideal number of credit hours.

3.2.4.3.4 Functional Requirement 4

The scheduler shall not generate a schedule that includes a number of credit hours greater than the desired (or default) number of credit hours plus one.

3.2.4.3.5 Functional Requirement 5

If the user provides no course load preference, a default of 15 hours shall be used.

3.2.4.3.6 Functional Requirement 6

The schedule shall be displayed on the profile page.

3.2.4.3.7 Functional Requirement 7

The user shall be redirected to the profile page upon scheduler completion.

3.2.4.3.8 Functional Requirement 8

The scheduler shall not schedule classes that have already been taken by the user.

3.2.4.3.9 Functional Requirement 9

The scheduler shall not schedule classes that conflict in times.

3.2.4.3.10 Functional Requirement 10

The scheduler shall not schedule additional classes for co-op semesters.

3.2.4.3.11 Functional Requirement 11

The scheduler shall not schedule classes for which the user has not completed the necessary prerequisite courses.

3.2.4.3.12 Functional Requirement 12

The scheduler shall ensure corequisite rules are followed.

3.3 Performance Requirements

- 95% of schedule generations shall take no longer than 10 seconds.
- The system shall be able to store profiles for at least 2500 users.
- The web page shall be able to handle 50 concurrent users.

3.4 Design Constraints

Due to time limits and hosting constraints, the myADVISE web application is forced to be smaller in scope. Firstly, the projected majors are being limited to the Speed School of Engineering instead of the entirety of the University of Louisville. The application could be expanded to handle other disciplines in the future. Secondly, the free hosting option also limits how much data can be stored, as well as having a non 24/7 uptime. The current hosting scheme also severely limits available bandwidth, hampering our ability to handle requests between multiple users concurrently. These constraints cause caching and other real time storage techniques to be too risky for the current minimally viable product. The application is also constrained by the amount of course information published by the University of Louisville; it can only rely on data that is currently available. The database limitations on row numbers also greatly affects how databases are designed; limits are based on rows, so as data as possible will be stored in one row.

3.5 Software System Attributes

3.5.1 Reliability

The program will:

1) make use of efficient queries to trim response and generation time
2) be built on readily scalable systems to meet users' needs
3) utilize data source efficiently to promote prompt response from web host
4) have reliable software interfaces to ease the user experience

3.5.2 Availability

1) The server host limits uptime to 20 hours a day
2) Capable of be scaled on demand based on traffic
3) Portions of data will be backed up and be capable of being restored in the event of an outage/failure

3.5.3 Security

1) The Django framework shall be used to ensure all website services are secure.

4.1 Table of Contents