Software Requirements Specification

for

myKSU Mobile Application

Version 1.0 Approved

Prepared by:

Brandon Merck: ***Coach, Developer***
bmerck@students.kennesaw.edu

Alix Teschner: ***Developer, Tester***
ateschne@students.kennesaw.edu

Ethan Dillon: ***Developer, Tester***
edillon7@students.kennesaw.edu

Joshua Gregory: ***Developer, Tester***
jgrego42@students.kennesaw.edu

24 Mar. 2025
SWE3623 W05

Table of Contents

1. Introduction [Ethan Dillon]

Kennesaw State University has been using the Owl Express website for years to allow students to register for classes, view their schedules, and pay their tuition all in one place. But despite more and more colleges creating mobile apps to allow students to view and interact with this information on the go, KSU does not have a mobile app equivalent to Owl Express. And as the next generation continues to be more dependent on their mobile devices, it is becoming more necessary to have an app equivalent for every internet service. And while Owl Express is a functional and helpful tool, its design is not well suited for mobile viewing, which is especially important for things as important as registering for classes or making tuition payments. But not only does the lack of an Owl Express app make KSU fall behind technologically, but it makes it harder for students in an increasingly mobile world. College students lead very busy lives, and because of this it is easier to miss important deadlines like registration times and tuition due dates, especially with young people who do not use or check their email as much as previous generations.

2. Overall Description [Joshua Gregory]

2.1. User Classes and Characteristics

- STUDENTS
    - This class consists of myKSU's primary users.
    - Student users are identified by KSU IDs, assigned by the existing University Student Information System (USIS). In addition to user-submitted myKSU profile data, the application will store the following USIS Student data (accessed via KSU ID):
        - Contact information
        - Class standing (freshman to graduate)
        - Academic standing
        - Course history and calculated GPA
        - Single or dual major, degree, or certificate
        - Optional minor or concentration
        - Financial data (aid, balances, and payment history)
    - Expect a secure, intuitive, mobile-friendly interface with quick access to academic planning and registration tools.

- ADMINISTRATOR FACULTY
    - Will manage course offerings (manually if needed) and enrollment limits, assist students with application usage, and other administrative domain tasks, per university policies.
    - Will require access to student records and account data stored on myKSU's database and the ability to manage course availabilities and enroll and unenroll courses.
    - Administrators will require training in Student user experience and myKSU Student Management Console. Appointing a representative from each administrative team to receive more extensive training and serve as subject matter expert and ambassador is recommended to limit training costs.

- IT FACULTY
    - Will be primarily responsible for integration, maintenance, security, and technical support for the myKSU system.
    - Will require backend access to manage database integrity, troubleshoot issues, implement updates, and all other system tasks.
    - Will ensure compliance with institutional IT policies and data privacy standards.
    - Interfacing between myKSU and University Information Technology System (UITS) will enable issue tickets to be submitted to UITS helpdesk from myKSU student, administrative faculty, and business management UIs.

- BUSINESS MANAGEMENT
    - Will oversee operational and financial aspects of myKSU, including resource allocation and various analyses.
    - Will require access to analytics and reporting tools to evaluate:
        - Cost management and performance
        - Application adoption and usage

        - Success metrics and goals
        - Overall return on investment
    - Will ensure application remains aligned with business goals, resource constraints, and university policies.

2.2. Operating Environment

The myKSU mobile application will operate within KSU's academic and administrative ecosystem with a mission to support students and faculty with course registration and academic planning. The system's backend infrastructure will be hosted on on-premises servers at KSU's main campus in Kennesaw, GA, where it will be integrated with existing university systems.

- Platforms and Hardware
    - Smartphone mobile clients
    - Windows machine desktop clients
- Architectures:
    - ARM64
    - x86-64
- Server Infrastructure:
    - On-premises data servers at Kennesaw State University, Kennesaw, GA
- Operating Systems and Versions
    - Mobile:
        - Android, v. 10+
        - iOS, v. 14+
    - Desktop:
        - Windows 10 and 11, 64-bit, v. 21H2+
- Server Environment
    - Windows Server
- Database Management System (DBMS)
    - Microsoft SQL Server and SQL Server Management Studio (SSMS)
- Location
    - Users: primarily US, South-East, Georgia (on- and off-campus)
    - Servers: Kennesaw State University, Kennesaw, GA
- System Integrations and Compatibilities
    - University Student Information System (SIS)
    - University Technology Information System (UTIS)
    - D2L Brightspace
    - Identity Management, Authentication, SSO Services
    - Owl Express Campus Portal
    - Financial & Billing Systems
    - Microsoft Office Suite and Power B.I.
    - Analytics and data analyses tools

3. System Features

3.1. Create Account & Login

3.1.1. Description

When a student uses myKSU for the first time, they will be prompted to enter their KSU ID and student credentials. This will be used to verify the student's identity. Successful submission will enable myKSU to interface with KSU's University Student Information System (USIS) to retrieve a range of student data for initial account setup. In addition to this, records will be created for the user-submitted myKSU user profile, geolocation data during use, and device data.

Main relation for a Student entity instance in myKSU's database:

STUDENT
Contact Information
Admission Date
Enrollment Status
Class Standing
Academic Standing
Calculated GPA
Course History with Final Grades
Major, Degree, or Program of Study
Minor or Concentration
Payment Activities (per semester)
User-Submitted myKSU Profile
Device Data
Geolocation Data

Following initial account setup and automated database operations, myKSU will become the primary manner method for updating USIS data, though the existing web-based campus portal application Owl Express will remain available as well. Upon system log-out via application close or inactivity, myKSU will interface with USIS to ensure data integrity is maintained across both systems.

Priority: HIGH

3.1.2. Stimulus/Response Sequences for Normal Flow

Action	*Response*
User successfully submits KSU ID and student credentials.	System interfaces with USIS to retrieve data needed to create a user account. System creates a main relation (account) for the student. System loads myKSU's main menu.
User unsuccessfully submits KSU ID and student credentials.	System displays an error message. System resets form inputs.
User selects "Kennesaw State University".	System opens KSU's homepage in a new tab in the default browser.

User selects “Contact Support”.	System opens KSU’s UTIS page in a new tab in the default browser.
User selects “myCourses”.	System navigates to the myCourses semester select screen.
User selects “myRegistration”.	System navigates to myRegistration semester select screen.
User selects “myAdvisor”.	System navigates to the digital advisory chat interface screen.
User selects “myPayments”.	System navigates to the myPayments semester select screen.
User selects “myProfile”.	System navigates to the user profile screen.

3.1.3. Functional Requirements

3.1.3.1. When myKSU is opened for the first time on a device, the system must prompt the user to enter their KSU ID and school credentials.

3.1.3.2. Whenever the KSU ID and school credentials are invalid, the system must inform the user and reset the form inputs.

3.1.3.3. The system must use the KSU ID and credentials to verify the student’s identity.

3.1.3.4. The system must interface with USIS to retrieve the verified student’s data: contact information; admission date; class standing; academic standing; current GPA; course history with final grades; major, degree, or program of study; minor and/or concentration; and payment history.

3.1.3.5. The system must create a user account by creating a main database relation with other relations to store USIS data.

3.1.3.6. The system must create a field in the main relation to store myKSU user-submitted user profile (initially set to null), geolocation data, and device data.

3.1.3.7. The system must store geolocation data and device data each time the user opens the application.

3.1.3.8. The system must confirm the device data is the same as the previously stored device data.

3.1.3.9. If the device data is not the same as the previously stored data, the system must prompt the user for their KSU ID and login information to confirm their identity.

3.1.3.10. The system must navigate to the correct screen when users interact with the UI.

3.2. View Courses by Semester & Unenroll Current Courses [Joshua Gregory]

3.2.1. Description

myKSU will allow users to view courses for each semester the student has been enrolled and for which USIS has data. Users can click each course to view course catalog information. For current semesters, users can unenroll the course from this screen.

Priority: HIGH

3.2.2. Stimulus/Response Sequences for Normal Flow

Action	*Response*

User selects the Home icon.	System navigates to main menu.
User selects a semester by clicking the "View semester".	System loads and displays enrolled courses for the semester.
User selects "Change semester."	System returns to semester select screen.
User selects a course.	System loads and displays course info.
User selects "Unenroll course".	System prompts the user to confirm.
User confirms the action.	System deletes the course from the user's account. System updates USIS.
User cancels the unenroll action.	System makes no changes. System removes the confirmation.

3.2.3. Functional Requirements

3.2.3.1. System must have access to the USIS.

3.2.3.2. System must have access to the school's information database (including course information).

3.2.3.3. If a user clicks on the home icon, system must redirect the user to the home page.

3.2.3.4. When a user selects their desired semester to view, system must access the USIS to acquire the currently selected courses student has enrolled into.

3.2.3.5. System must display enrolled courses after the "View semester" button is selected.

3.2.3.6. The currently displayed semester courses must be shown on a separate page.

3.2.3.7. The "Change semester" button must exist on the screen currently displaying enrolled courses.

3.2.3.8. When a user selects the "Change semester" button, system should return user to the "View semester" screen.

3.2.3.9. When a user selects a certain course on the viewing screen, system must acquire all relevant information on the course.

3.2.3.10. Selected courses must display relevant information on the course. (Date, time, seats remaining, professor name, campus)

3.2.3.11. When a course is selected, a button to allow the user to unenroll from the course should appear.

3.2.3.12. If the user chooses to unenroll from a selected course, a prompt asking the user if they are sure should appear.

3.2.3.13. Should the user select "Yes" on the 'are-you-sure- page, system should proceed with unenrollment process.

3.2.3.14. Should the user select "No" on the 'are-you-sure' page, system should return user to previous viewing screen.

3.2.3.15. Regardless of the selected button on the 'are-you-sure' page, the window should be removed after button is selected.

3.2.3.16. If the student has confirmed the unenrollment of a selected course, the system should update the school's course database with the removal of the student from the course.

3.2.3.17. If the user unenrolls from a course, the system should provide visual feedback that they have successfully, or unsuccessfully, been removed from the course.

3.2.3.18. Any course that has been unenrolled from successfully should be removed from the student's record.

3.2.3.19. If a user closes the app whilst in the middle of a process, (such as unenrolling from a course), the next time the app is opened up, redirect the user to the Home screen.

3.2.3.20. All unsaved changes should be removed if user closes app prematurely.

3.3. Search Catalog, View Course Recommendations, Enroll Course [Alix Teschner, Brandon Merck]

3.3.1. Description

Students will be able to search for all available courses at KSU using course names, numbers, and subjects. They will also be able to view courses specific to their major. The system will be able to show them course recommendations-based on their major and what courses they logically should be taking next depending on what they are currently taking. They can then use this to enroll in the courses they would like to take. Students can also choose to be on the waitlist of a course if it is open. They will be able to choose what semester they would like to view courses for and enroll in.

Priority: HIGH

3.3.2. Stimulus/Response Sequences for Normal Flow

Action	*Response*
User selects the “Home” icon	System navigates to main menu
User selects a semester	System loads a menu page that allows them to search for available courses or view their recommended courses for that semester.
User chooses to type into search bar	System uses autofill to help the user find the course they are trying to search for
User uses dropdown menu to search	System drops a menu with all available courses in alphabetical order
User selects a class	System loads a page with all available sections for that class
User selects a section to enroll in	(If class is open) System prompts the user to confirm they would like to enroll (If class has a waitlist) System informs user the class is full and prompts the user to confirm they would like to be on the waitlist (If class and waitlist are full) System informs user they cannot enroll in the class
User confirms the enrollment	(If enrollment is successful) System adds the class to the user’s account (If enrollment is unsuccessful) System informs user that they are unable to enroll in the class and informs them of what holds are causing the action to be incomplete System updates USIS
User cancels the enrollment	System makes no changes

	System removes confirmation prompt
User confirms the waitlist option	System adds the user to the class's waitlist System updates USIS
User cancels the waitlist option	System makes no changes System removes confirmation prompt
User selects recommended courses	System loads a page of recommended courses based on the user's major and classes already or currently being taken

3.3.3. Functional Requirements

3.3.3.1. System must have access to the USIS.

3.3.3.2. System must have access to the school's information database (including course information).

3.3.3.3. If a user clicks on the home icon, system must redirect the user to the home page.

3.3.3.4. When a user selects a semester, system should retrieve courses from school's information database.

3.3.3.5. System must populate a search bar to search for courses.

3.3.3.6. If a user begins typing in the search bar, the system should use a predictive text mechanism to auto-populate course names that the user is likely looking for.

3.3.3.7. The courses the system is trying to predict the user is typing should appear in a drop-down menu under the search bar.

3.3.3.8. The options in the drop-down menu should be buttons that can be selected.

3.3.3.9. The options in the drop-down menu should appear in alphabetical order.

3.3.3.10. A button on the side of the search bar should exist that shows all recommended courses based on major.

3.3.3.11. Should a user click on this button, system must access all courses related to their particular major.

3.3.3.12. If recommended courses based on major button is selected, do not display any courses in search bar drop-down box not relating to major.

3.3.3.13. When a user selects an option in the drop-down menu, the system must redirect the user to the corresponding course page.

3.3.3.14. If a user types something in the search bar that does not exist in the database, the drop-down menu must be blank.

3.3.3.15. When the course page loads, the system must access all courses with the same name and display them on the page.

3.3.3.16. Each course displayed must show all relevant information pertaining to the course. (Course number, course name, professor name, seats available, time, date, and description)

3.3.3.17. Every course displayed must have a button under its listing that says "Enroll".

3.3.3.18. If a student clicks on the "Enroll" button for a certain course, and the course has seats open still, system must prompt the user to confirm if they'd like to enroll in this course, and display the course's information.

3.3.3.19. If the user selects "Yes" to confirm enrollment, system must update that course's information to reflect a student registering for that class.

3.3.3.20. If the user selects "No" to confirm enrollment, return user to previous view courses screen.

3.3.3.21. If a student clicks on the "Enroll" button for a certain course, and the course has no seats open, the system must prompt the user that there are no more seats available.
3.3.3.22. Should no seats be available after attempting to register for a class, and there are seats remaining in the waitlist, a prompt asking the user if they want to join a waitlist should appear.
3.3.3.23. If the user says "Yes" to join the waitlist, system must add user to the waitlist.
3.3.3.24. If the user says "No" to join the waitlist, system should return them to previous view course screen.
3.3.3.25. Should no seats be available after attempting to register for a class, and there are no seats remaining in the waitlist, a prompt telling the user this course cannot be registered for should appear.
3.3.3.26. Under the prompt telling the user the class couldn't be registered for, there should be a button that says "Return".
3.3.3.27. If the user presses the "Return" button under the failed registration prompt, return to previous view course screen.
3.3.3.28. If a user attempts to enroll in a course that has open seats, but the user has holds on their account, system should prompt the user that they cannot register for course because they have holds on their account preventing them.
3.3.3.29. Under the prompt advising the user they have holds on their account, have a button that says "Return".
3.3.3.30. If the user presses the "Return" button under the prompt telling the user they have holds on their account, return to previous view course screen.
3.3.3.31. If a user closes the app whilst in the middle of registering for classes, remove all changes.

3.4. View and Export Digital Schedule [Alix Teschner]

3.4.1. Description

The digital schedule will allow students to view the schedule for the classes they are enrolled in. They will also be able to create a hypothetical schedule using available classes for the semester they select. This will allow them for insight on what classes and sections they should be taking to give them their preferred schedule. Students will then be able to export this schedule as a PDF or PNG to either save to their files or print for their convenience,

Priority: MEDIUM

3.4.2. Stimulus/Response Sequences for Normal Flow

Action	*Response*
User selects the "home" icon	System navigates to the main menu
User selects a semester	(If user is not yet enrolled in classes) System loads a blank schedule for the selected semester (If user is enrolled in classes) System loads a schedule of the classes the user is taking

User searches for a class	System loads a list of all available sections for that class
User selects a class section to add onto the schedule	System adds that class to the schedule at the specific days and times that section has class
User selects delete button next to an added class	System deleted that class from the schedule
User selects the clear button next to the schedule	System prompts user to confirm that they want to clear the schedule
User confirms the deletion	System clears the schedule
User cancels the deletion	System makes no changes to the schedule
User selects a class that conflicts with another class already added	System informs user that the selected class conflicts with an already added class and prompts them to conform if they would like to continue
User confirms class selection	System adds conflicting class to schedule, leaving an error message next to the two or more classes that overlap
User cancels the class selection	System cancels addition of the class and returns to schedule
User selects the export button	System prompts user to choose what format they would like to export the schedule as
User selects their chosen format	System prompts user to save schedule to their device or to print the schedule
User selects save	System brings up user's file system to name and save the file
User selects print	System brings up user's print function to print the schedule
User cancels export	System cancels export and returns to schedule
User saves schedule	System saves schedule to its database

3.4.3. Functional Requirements

3.4.3.1. System must have access to the USIS

3.4.3.2. System must have access to the school's class information database

3.4.3.3. If the user clicks the home button, the system must redirect them to the home page

3.4.3.4. When the user selects a semester, the system should retrieve courses available that semester from the school's database. If the user is already enrolled in courses, the system should retrieve the courses they are enrolled in using the USIS.

3.4.3.5. If the user is already enrolled in courses, the system should access the USIS to create a digital schedule of those courses.

3.4.3.6. If the user enrolls or unenrolls from a class, the system must update the digital schedule accordingly.

3.4.3.7. System must populate a search bar to search for available courses.

3.4.3.8. If a user begins typing in the search bar, the system should use a predictive text mechanism to auto-populate course names that the user is likely looking for.

3.4.3.9. The courses the system is trying to predict the user is typing should appear in a drop-down menu under the search bar.
3.4.3.10. The options in the drop-down menu should be buttons that can be selected.
3.4.3.11. The options in the drop-down menu should appear in alphabetical order.
3.4.3.12. System must be able to add any available course from the school's database to the user's schedule.
3.4.3.13. System must be able to inform the user when there are schedule conflicts.
3.4.3.14. System must prompt user to confirm they would like to continue when added a course with a conflicting time frame.
3.4.3.15. System must present an error message next to overlapping courses.
3.4.3.16. System must allow the user to create their own schedule using available courses in the school's database.
3.4.3.17. System must be able to add or delete courses from the schedule based on the user's instruction.
3.4.3.18. System must prompt user to confirm when clearing or deleting a schedule.
3.4.3.19. System must be able to save any built schedules to its database if prompted by the user.
3.4.3.20. System must be able to export the digital schedule as a PDF or PNG. This file must then be able to be saved to the user's device or printed.

3.5. Use Digital Advisor Chat Interface [Brandon Merck]

3.5.1. Description

Students will be provided with an AI advisor that reads student information and is able to provide meaningful feedback on the next steps the student should take. The AI should have access to the information inside the University Student Information System (USIS) as well as the classes listed inside the school database.

This will act as a simple chatbot that can read and answer simple questions and should have many helpful capabilities included for the student's convenience. These abilities include creating a class schedule, allow to add and drop classes through the bot, give payment information including tuition costs, answer what pre-requisites are required for certain classes, give students their GPA on request, declare a major for a student ready to declare, ask about faculty office hours, view number of currently registered credit hours, return previously completed, dropped and failed courses, advise a student if they are eligible for a certain class if requested, give course information such as days of the week to attend, and project estimated graduation date.

The bot should also give general help for navigating the app. Such help should include instructions on how to register for classes, how to set up payment, what steps need to be taken for setting preferences, and be able to contact support for more questions.

Priority: HIGH

3.5.2. Stimulus/Response Sequences for Normal Flow

(Note: The italicized segments represent chatbot responses, words inside square brackets [] represent variable names, and the description next to the + represents the system response to the user instruction.**)**

Action	*Response*
User requests their GPA.	*"Your current GPA is [currentGPA]"* +System returns current student GPA
Eligible user requests to register for a class by course number.	*"You are now registered for [courseName] in the [registeredSemester] semester!* +System registers current student for specific course using input course number
User requests to be dropped from a particular course.	*"You have successfully been removed from the course [courseName]"* +System removes the student from registration of particular course
User asks to have a class schedule made for next semester.	*"According to your declared major [declaredMajor], you should register for [courseName], [courseName], [courseName], and [courseName]."* +System checks student major, number of credit hours they want to take per semester, and preferred campus, and generate a list depending on all these variables.
User asks how much their tuition will cost.	*"You owe $[studentTuition] in tuition fees this semester."* +System should access user tuition and return the current value.
User asks what the prerequisites to a particular course are.	*"The prerequisites to the course [courseName] are: [prerequisite], [prerequisite], and [prerequisite]."* +System should find prerequisites and return all prerequisites in an understandable format.
User requests to be declared as a particular major.	*"You have successfully declared yourself as a(n) [currentMajor] major!"* +System should update user status in USIS to represent new declared major.
User requests office hours of a particular department or professor.	*"The current office hours of [department]/[professor] are: [officeHoursArray]."* +The system should find the entity the user is looking for, determine their available

	office hours, and return the office hours in a neat array showing Mon-Sun hours.
User asks how many current credit hours they are registered for.	*"You are currently enrolled in [numOfCourses] courses and are registered for [numOfCreditHours] credit hours."* +System validates current user's credit hours and return it in readable format.
User requests to view previously taken courses.	*"Here are the courses you've already taken: [previouslyTakenCoursesArray]"* +The system should acquire the student's prior history, retrieve if the student passed, withdrew, withdrew-failed, or failed said course, and return all courses in an organized array for viewing.
User asks for details about a particular course.	*"Here are the details of the course [courseName]: [courseInformation]"* +System will retrieve all relevant information about particular course, including days and times of the week course is in session, credit hours, number of seats left, and prerequisites, and return all this information in a readable array for viewing.
User asks when they are estimated to graduate.	*"Based on the number of credit hours you want to take per semester, you are estimated to graduate on the [estimatedGraduationDay]."* +System should retrieve user remaining courses, tally all remaining credit hours, divide that by current user's preference for credit hours per semester, and give an estimated date of graduation.
User asks if they have met the requirements for a particular course.	*"Based on the classes you've already taken, you are (are not) eligible for [courseName]."* +System determines if all prerequisites for particular course are met or not, and returns with a straight forward yes or no. If not, return all courses still needed.
User asks for help on registering for classes.	*"Here are the steps for registering for classes in the myKSU app: [registrationInstructionsArray]"*

	+System should provide the user registration instructions in a presentable manner.
User asks how to set up payment.	*"Here are the steps for setting up payment in the myKSU app: [paymentSetupInstructionsArray]"* +System should provide the user payment setup instructions in a presentable manner.
User asks how to set preferences.	*"Here are the steps for setting semester preferences in the myKSU app: [preferenceInstructionsArray]"* +System should provide the user preference instructions in a presentable manner.
User asks to contact support.	*"To reach out to customer support, please use the following: [supportEmail] [supportPhoneNum] [supportAddress]"* +System should provide the user support information in a presentable manner.
User requests something the system does not recognize how to answer.	*"I'm sorry, I do not understand how to answer that.* *To reach out to customer support, please use the following: [supportEmail] [supportPhoneNum] [supportAddress]"* +System will tell the user it does not understand, and should provide the customer support information in a presentable manner.

3.5.3. Functional Requirements

3.5.3.1. When an authenticated user logs into the app, the virtual advisor button must appear.

3.5.3.2. When a user interacts with the virtual advisor button, the chat window must appear.

3.5.3.3. The virtual advisor must have access to the USIS at all times.

3.5.3.4. The virtual advisor must have access to the school's courses database.

3.5.3.5. When the user types a prompt into the chat window, a response from the bot must be generated.

3.5.3.6. If the bot is unable to understand a question or prompt, it must return to the user a response addressing its inability to understand.

3.5.3.7. If the bot is unable to understand a question or prompt, it must return to the user a list of customer support contact information.

3.5.3.8. When a student requests to register for a class using course numbers, the bot must be able to locate said course.

3.5.3.9. When a student requests to register for a class, the bot must verify that said class has seats available.

3.5.3.10. When a student requests to register for a class by a course name, the bot must be able to find the courses with that name.

3.5.3.11. When a student requests to register for a class, the bot must verify that all prerequisites have already been met.

3.5.3.12. When a student requests to register for a class, the bot must verify there are no holds on the student's account.

3.5.3.13. When a student requests to register for a class, if nothing is hindering that student after all checks are made, the bot must register the student for the requested class.

3.5.3.14. A user who requests their GPA must have their GPA returned in a readable format.

3.5.3.15. If a user requests to view all previous courses, the system must access the student's prior course records.

3.5.3.16. If a user requests to view all previous courses, the system should display all previous courses in a neat array.

3.5.3.17. When displaying the array of previously taken courses, the format must display if each class was passed, withdrawal, withdrawal-failed, or failed.

3.5.3.18. If a user requests to be removed from a course, the bot shall remove the student from the course database.

3.5.3.19. The bot should inform the user of the success or the failure to remove them from the course.

3.5.3.20. If a user requests to see their tuition fees, the bot must access the student's semester tuition fees.

3.5.3.21. When the fees have been located, the bot should display the tuition fees to the user in the chat box.

3.5.3.22. If a user asks to have a course schedule made, the bot should access the USIS to verify selected major.

3.5.3.23. Once the selected major has been located, the bot must determine all incomplete prerequisites for graduating in order.

3.5.3.24. Once incomplete prerequisite classes are located, the bot should generate a list of classes still needed to graduate.

3.5.3.25. Of the generated list, the bot must choose courses with availability.

3.5.3.26. Once the list is refined to courses with availability, the bot should create a course schedule based on the user's preferred number of credit hours desired to be taken per semester.

3.5.3.27. If a user asks what the prerequisites are to a particular course, the bot should access the school's course information.

3.5.3.28. From the course information, the bot should locate all prerequisite classes for said course.

3.5.3.29. Once prerequisite classes are found, return to the user in the box a string describing the courses still needing to be taken.

3.5.3.30. If a user tells the bot to be declared as a certain major, the bot must update the major status of the student in the USIS.

3.5.3.31. When the status of the student is changed in the USIS, the bot must give verbal feedback in the chat box for the user reiterating what major they selected.

3.5.3.32. If a user requests for the office hours of a particular department, the bot should locate the listed hours inside the school's information database.
3.5.3.33. Return to the user the discovered office hours of the selected department via a string in the chat box.
3.5.3.34. If a user requests for the office hours of a particular professor, the bot must locate the professor's courses information in the school's information database.
3.5.3.35. Return to the user the discovered office hours of the selected professor via a string in the chat box.
3.5.3.36. If a user asks how many credit hours they are currently enrolled for, the bot must locate their current registered office hour number.
3.5.3.37. Bot must return located currently registered office hours via a string in the chat box.
3.5.3.38. If a user prompts the bot to give information on a particular course, the bot must locate all the available information on said course through the school's information database.
3.5.3.39. Once the available information on a course being requested has been acquired, return to the user the information through the chat bot text box.
3.5.3.40. The returned information on a course should include course credit hours.
3.5.3.41. The returned information on a course should include course days of the week in session.
3.5.3.42. The returned information on a course should include course hours class is in session.
3.5.3.43. The returned information on a course should include number of seats still available.
3.5.3.44. The returned information on a course should include prerequisite classes.
3.5.3.45. If a user asks for an estimated date of graduation, the bot must consider student declared major.
3.5.3.46. Once declared major has been considered, all courses the student has not taken must have their credit hours summed together.
3.5.3.47. When all of the remaining courses have their hours summed together, the bot should divide remaining credit hours by the current user's preferred number of credit hours they want to take per semester.
3.5.3.48. The bot must take the answer to the divided number of hours and tell the user that they have about that many more semesters before they graduate in the chat bot text box.
3.5.3.49. If a user asks if they have met the requirements to register for a particular course, the bot must locate the current course in question's prerequisite classes.
3.5.3.50. Once the prerequisite classes have been located, the bot must compare the prerequisite classes with the current user's list of previously taken courses.
3.5.3.51. If all courses have already been passed, the bot must return to the user that they are eligible to register for said course in the chat box.
3.5.3.52. If not all of the courses have been passed/taken, the bot must return to the user that they need to take a few more classes before they can register in the chat box.
3.5.3.53. The bot will display to the user in the chat box the remaining courses they need to take before they can take their requested course.
3.5.3.54. If the user asks for help on registering for classes, the bot should return to the user myKSU's registration instructions through the chat bot text box.
3.5.3.55. If the user asks for help on setting up their payment information, the bot should return to the user myKSU's payment instructions through the chat bot text box.

3.5.3.56. If the user asks for help on setting their course preferences, the bot should return to the user myKSU's preferences instructions through the chat bot text box.

3.5.3.57. If the user asks to contact support, the bot must display through the chat bot text box a list of customer support contact information.

3.6. View Payment Activity by Semester & Pay Outstanding Fees [Ethan Dillon]

3.6.1. Description

This feature allows students to view the entire history of payments they have made to KSU as well as allowing students to view any outstanding dues and make payments on them. The system will allow students to pay with their credit card or use third-party services such as PayPal, Apple Pay, or Google Pay.

Priority: HIGH

3.6.2. Stimulus/Response Sequences for Normal Flow

Action	*Response*
User navigates to "payment" page	The system shows the student their recent payment activity from the past month and a summary of the current amount due and the date of when each amount is due. There is an option to view past payments and an option to make a payment.
User clicks on "make payment" button underneath "current amount due" section.	The system shows the user the current balances due with the date of when that is due as well as a grand total amount due underneath these. There is an option to select an amount from the list or to input the amount manually. Then there is an option to checkout at the bottom.
User selects all amounts due and clicks "checkout".	The system displays labeled text fields to input credit card information as well as links to pay with Paypal, Apple Pay, and Google Pay. There is an option to submit credit card information or to cancel transaction.
User enters credit card information and presses "confirm payment".	The system checks that the credit card information was entered correctly then shows the user a confirmation message stating that their payment information has been received and is being processed and shows them a digital receipt that has an option to print or save.
User presses "Print" on the receipt.	The system opens the operating system's printing feature with this document.

User presses “cancel” on the payment screen.	The system returns the user to the home screen and discards the inputted information.

3.6.3. Functional Requirements

3.6.3.1. The system must allow users to view their recent payment activity from the past month as well as a summary of their current payments due upon accessing the “payments” page.
3.6.3.2. The system must allow users to expand their payment history information for a specific payment when selected.
3.6.3.3. The system must allow users to view their payment history for their entire time as a KSU student.
3.6.3.4. The system must allow users to view a detailed account of the payments due that includes the due date and amount due when selecting the “make payment” button.
3.6.3.5. The system shall provide a “pay now” button option to make a payment on the due payment that they are currently viewing.
3.6.3.6. The system shall allow users to input the amount they want to pay within two decimals on the “choose payment amount” screen.
3.6.3.7. The system shall not allow users to input an amount greater than the total amount listed on the “choose payment amount” screen.
3.6.3.8. The system shall allow the user to input credit card information or to select a third-party payment method inside the “make payment” window.
3.6.3.9. The system shall have a “cancel” option in the “make payment” window.
3.6.3.10. The system shall have a “confirm payment” option in the “make payment” window.
3.6.3.11. The system shall have a “cancel” option in the “select payment amount” window.
3.6.3.12. The system shall have a “continue” option in the “select payment amount” window.
3.6.3.13. The system shall validate payment details such as correct credit card number length and expiration date before allowing the user to confirm payment.
3.6.3.14. The system shall confirm payment is received through a confirmation message.
3.6.3.15. The system shall give the option to cancel the current payment before it is submitted.
3.6.3.16. The system displays currently pending payments in the payment history section.
3.6.3.17. The system shall update the amount due or remove the amount due entirely based on how much the user paid.
3.6.3.18. The system must only allow authorized users to access payment history and information.
3.6.3.19. The system will display a digital receipt after the credit card information has been received to confirm payment transaction.
3.6.3.20. The system shall allow for the export of the digital receipt in pdf format.
3.6.3.21. The system shall allow for the printing of the digital receipt.

3.7. View and Edit User Profile [Ethan Dillon]

3.7.1. Description

This feature allows users to view their account information such as their name, personal and school email, KSU ID, and account bio. This feature also allows users to edit some of their account information such as their personal email and bio. To edit the sensitive information requires an administration account.

Priority: LOW

3.7.2. Stimulus/Response Sequences for Normal Flow

Action	*Response*
User navigates to profile page.	System shows the users profile information (student ID image, name, emails, KSU ID, bio).
User clicks “edit” button	System displays editable prefilled input boxes for data that can be changed.
User updates input boxes with new information.	System confirms valid input.
User clicks “save” button.	System pushes the new data to the database and shows a success message confirming the save.
User clicks “cancel” button	System discards new data and returns to profile page.

3.7.3. Functional Requirements

3.7.3.1. The system must allow users to view their profile ID picture, name, student email, personal email, KSU ID number, and bio upon accessing the “profile” page.

3.7.3.2. The system shall allow users to edit their personal email and bio through an edit screen that is accessible through an “edit” button on the profile screen.

3.7.3.3. The system must ensure only authorized users access user profiles so that each user only accesses their own profile.

3.7.3.4. The system shall ensure inputted email follows standard format before submitting edits and saving changes.

3.7.3.5. The system will save edits made to the database after the user clicks the “save” button.

3.7.3.6. The system shall confirm changes were saved by displaying a confirmation message upon submission of changes.

3.7.3.7. The system shall include a “cancel” button to cancel current edits and revert to previous values.

4. Quality Attributes

4.1. External Qualities [Alix Teschner]

- Availability:
  - myKSU shall achieve 99.9% uptime over 12 months.
- Performance:
  - myKSU shall take no longer than 45 seconds to display any menu.
- Useability:
  - myKSU shall be intuitive and easy to use with 99% of user testers rating the application as “easy to use”.

4.2. Internal Qualities [Brandon Merck]

- Integrity:
  - myKSU shall ensure that data integrity is maintained at-rest and in-transit.
- Security:
  - myKSU shall encrypt data in-transit and be safeguarded against malicious actors.
- Portability:
  - myKSU shall be useable for iOS and Android devices.

4.3. Attribute Prioritization [Joshua Gregory]

Priority		**Security**	**Integrity**	**Useability**	**Portability**	**Availability**	**Performance**
6	**Security**	=	<	<	<	<	<
5	**Integrity**	^	=	<	<	<	<
4	**Useability**	^	^	=	<	<	<
3	**Portability**	^	^	^	=	<	<
2	**Availability**	^	^	^	^	=	<
1	**Performance**	^	^	^	^	^	=

4.4. Other non-functional requirements as relevant [Ethan Dillon]

- **Responsiveness**:
  - The myKSU UI shall respond to user actions within 200 milliseconds under normal operating conditions (as measured during usability testing).
- **Accessibility**:
  - The myKSU UI shall comply with WCAG 2.1 Level AA standards.
- **UI Design**:
  - The myKSU UI shall incorporate KSU’s logo, color scheme, and typography.

5. Credits

1. Introduction: Ethan Dillon

2. Overall Description:
    2.1. Joshua Gregory
    2.2. Joshua Gregory

3. System Features:
    3.1. Joshua Gregory
    3.2. Joshua Gregory
    3.3. Alix Teschner & Brandon Merck
    3.4. Alix Teschner
    3.5. Brandon Merck
    3.6. Ethan Dillon
    3.7. Ethan Dillon

4. Other Non-Functional Requirements:
    4.1. Alix Teschner
    4.2. Brandon Merck
    4.3. Joshua Gregory
    4.4. Ethan Dillon