CS 3354 Software Engineering

Final Project Deliverable #2

Comet Dash

Group Members: Sophia Chau, Suchita Mamindla, Yara Abdelhadi, Kimberly Niemiec, Xander Corcoran, Hassan Hashemian, Bryan Macias, Ronan Rao

1: Delegation of Tasks

1. Sophia Chau: commit everything to GitHub, comparison with other projects, PowerPoint, dashboard view demo
2. Xander Corcoran: Cost, effort, pricing estimates, PowerPoint (styling), expanded view for a class demo
3. Hassan: Cost, effort, pricing estimates, Conclusion, PowerPoint, studyzone demo
4. Suchita Mamindla: Project Scheduling, Conclusion, PowerPoint, dashboard view demo
5. Bryan Macias: Test plan + test code, PowerPoint
6. Yara Abdelhadi: Cost, effort, pricing estimates, PowerPoint, studyzone demo
7. Kimberly Niemiec: Test plan + test code, PowerPoint
8. Ronan Rao: Test plan + test code, PowerPoint, expanded view for a class demo

2: Project Deliverable 1 content

2.1: Refined Project Proposal

1. Goals + Motivation
    a. Why we chose this project specifically: We want to develop a one-stop application for students at UT Dallas. This robust application will give them a holistic view of their classes, grades, assignments, study habits, etc; such an application will assist UTD students in preparing for their courses and remaining focused on their responsibilities throughout the semester.
    b. Where we expect our design to be used in real life: Students at UTD can use the platform to have a collective dashboard of elements that correspond with academic necessities. The system increases student engagement with platform specifics as all university information is easily accessible and provides the quick navigation aspect for users.
2. Delegated Tasks by Member
    a. Deliverable 1
        i. Sophia Chau: Github 1.4, Assumptions for the project, Pick software process model, List software requirements
        ii. Xander Corcoran: Github 1.2/1.3, Class Diagram, Architectural Diagram
        iii. Hassan Hashemian: Case Diagram, Sequence Diagram
        iv. Suchita Mamindla: Github 1.5, Assumptions for the project, Pick software process model, List software requirements
        v. Bryan Macias: Class diagram, Architectural design
        vi. Yara Abdelhadi: Case Diagram, Sequence Diagram
        vii. Kimberly Niemiec: Class Diagram, Architectural design
        viii. Ronan Rao: Case diagram, Sequence diagram
    b. Deliverable 2

i. Sophia Chau: commit everything to GitHub, comparison with other projects, PowerPoint, dashboard view demo
ii. Xander Corcoran: Cost, effort, pricing estimates, PowerPoint (styling), expanded view for a class demo
iii. Hassan: Cost, effort, pricing estimates, Conclusion, PowerPoint, studyzone demo
iv. Suchita Mamindla: Project Scheduling, Conclusion, PowerPoint, dashboard view demo
v. Bryan Macias: Test plan + test code, PowerPoint
vi. Yara Abdelhadi: Cost, effort, pricing estimates, PowerPoint, studyzone demo
vii. Kimberly Niemiec: Test plan + test code, PowerPoint
viii. Ronan Rao: Test plan + test code, PowerPoint, expanded view for a class demo

3. Proposal feedback: No issues to address from project proposal

2.2: Github Repository

Link: https://github.com/IXtimes/3345-softwareEngineeringBaddies

2.3: Delegation of Tasks

1. Sophia Chau: Github 1.4, Assumptions for the project, Pick software process model, List software requirements
2. Xander Corcoran: Github 1.2/1.3, Class Diagram, Architectural Diagram
3. Hassan: Case Diagram, Sequence Diagram
4. Suchita: Github 1.5, Assumptions for the project, Pick software process model, List software requirements
5. Bryan Macias: Class diagram, Architectural design
6. Yara Abdelhadi: Case Diagram, Sequence Diagram
7. Kimberly Niemiec: Class Diagram, Architectural design
8. Ronan Rao: Case diagram, Sequence diagram

2.4: Software Process Model + Assumptions

Incremental testing allows for development in stages providing a stronger framework to build new elements for the platform. This is particularly applicable to the Comet Dash project because Comet Dash has many features that are easy to compartmentalize for quicker development.

Using prototyping, data can be gathered from user interactions giving developers a stronger understanding of how the platform is utilized and how to optimize key elements. Prototypes also provide stakeholders with a clear vision of how the project will function, making them more likely to support the project moving forward. In a project like Comet Dash, which hinges on the cooperation of several different entities (e.g., the university, Nebula Labs, etc.), prototypes can help convince stakeholders that they should continue their involvement in the project.

Assumptions:

1. We are working with the collaboration of the university
    a. We have access to certain information that may not be publicly available(using integration software)
    b. We are integrating with UTD's system so that student registrations and other relevant information is automatically imported into their Comet Dash account
2. We are working with the permission and collaboration of the team that runs UTD grades and RMP so we have access to their API and information
3. There will be adequate money available to us for purchase of certain licenses that may be required for this project (e.g., database services we need to buy)
4. All data is integrated into the platform, but there is a layer of abstraction so that a third party who gains access to the system (either with permission or without permission of the developers) cannot access personally identifiable information or any other legally protected data
5. We have access to Blackboard/eLearning so that students can import their due dates into the Comet Dash

<u>2.5: Software Functional Requirements</u>

1. Student assignment reminders
    1.1. Student can choose to set weekend notifications
    1.2. Otherwise, students are notified of an assignment the working day before it is due (e.g., if an assignment is due at 5pm on Thursday, a student will be reminded on 9am Wednesday that they have an assignment due soon. Similarly, if an assignment is due 11:59 PM on Monday, a student will be notified at 9am on the Friday before about their upcoming assignment)
2. Syllabus collection
    2.1. All available syllabuses for a student's courses will be easily accessible from the Comet Dash website
    2.2. The user will be able to see a button to see their syllabuses on the dashboard, and from there they can choose from a menu of their classes which syllabus they wish to access
        2.2.1. Classes whose syllabuses are not yet accessible will be grayed out
    2.3. The syllabuses will be obtained from coursebook information

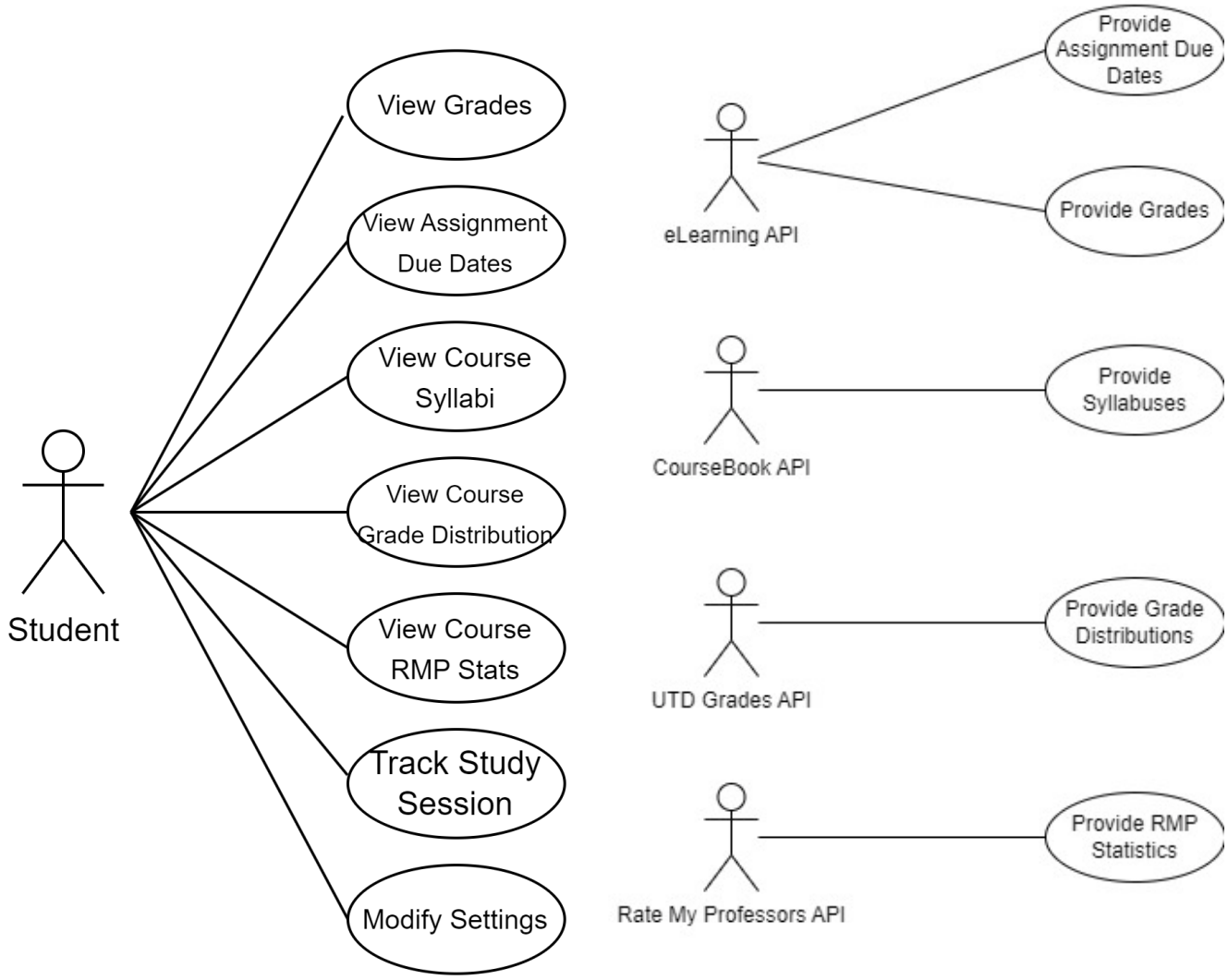
3. Grade distribution statistics
    3.1. The historic grade distributions of each class the student is enrolled in, taken from UTD grades
    3.2. For each class the student is taking with a particular professor, all of the previous grade distributions that are available will be averaged together
        3.2.1. Students will also be able to directly click on links to view all previous grade distributions on UTD grades
4. Grades for each class
    4.1. For each missing assignment, it is in a bright color so that the student can easily identify missing assignments
    4.2. The average grade, calculated from eLearning, is displayed for every class the student is currently registered
        4.2.1. For classes with no grades, the system will display N/A
    4.3. Students can click on particular classes to see a more detailed view of their grades (i.e., the individual grades for all assignments that have been graded)
5. Rate my professor statistics
    5.1. Professors for all enrolled classes and their RMP scoring will be easily accessible on the dashboard
    5.2. During class registration periods, professors who are in the "student shopping cart" will have RMP scorings listed in a gray outline
6. Study zone
    6.1. Student can allocate blocks of time for study sessions and use study platform links on the site (e.g., Quizlet, Notion, Obsidian, Udemy, Coursera)
    6.2. Students can record the time they have spent studying for each class, either categorized by assignment or categorized only by class
        6.2.1. Students can also record study times for exams and other major assignments
        6.2.2. Statistics on average time spent studying for each class will also be available to students
    6.3. Users will not be able to schedule study sessions in times that are blocked by classes
7. Dashboard view
    7.1. Cards of information displayed
        7.1.1. Current grade for each class, taken from eLearning
        7.1.2. Syllabi for classes
        7.1.3. Grade distributions + RMP averages
        7.1.4. Study time scheduling block and important study time statistics
        7.1.5. Upcoming or missing assignments
    7.2. A menu where you can access your profile, settings

7.2.1. Settings will contain notification settings, the ability to change weekend notifications, and the ability to hide certain views (e.g., if you do not want to see your grades on the dashboard, turn that off)
7.2.2. Settings can also modify the look of the website (i.e., darkmode)
7.2.3. Profile can be customized (profile picture, display name, etc.)

2.6: Non-Functional Requirements

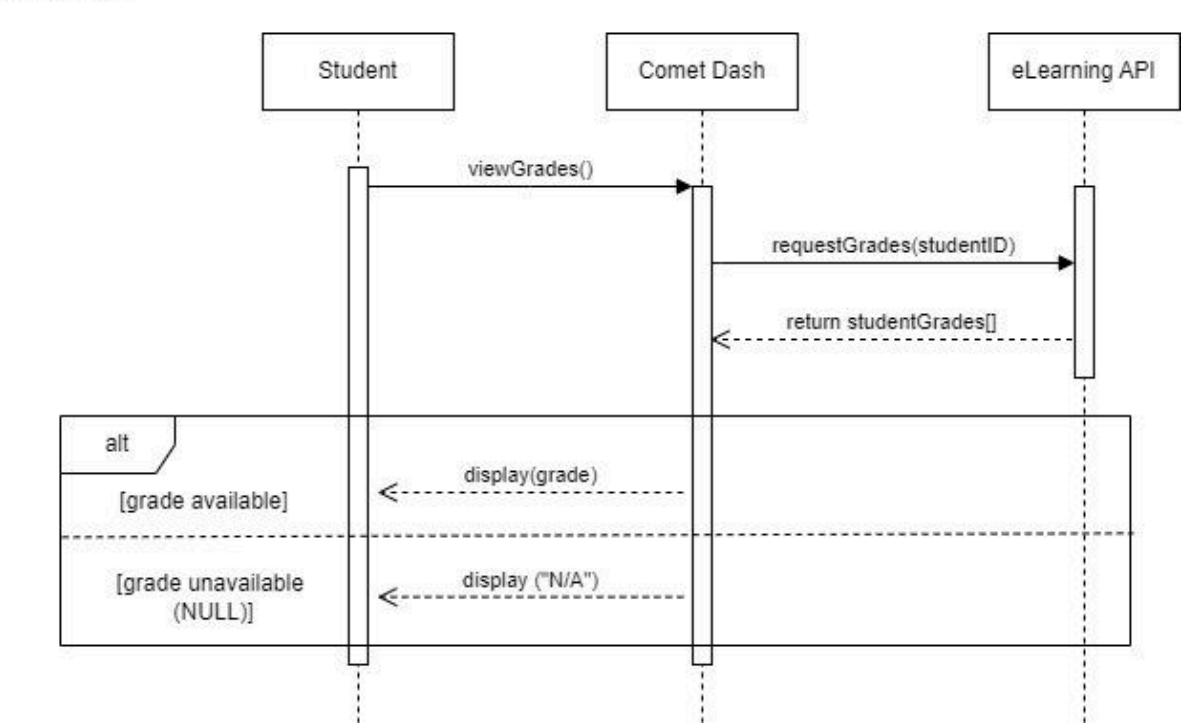
1. Product requirement: Security
    1.1. All parts of the system must be secured so that data and personally identifiable information is protected
    1.2. None of the data should be able to be accessed by an unauthorized third party
        1.2.1. Emphasis on security for PII
2. External requirement: Legal protections
    2.1. No student's data should be viewable by anyone but them (FERPA protections)
        2.1.1. Assumption: this website is under the regulation of the United States government
    2.2. If anyone's data is compromised, they should be notified of that breach as soon as it is found out
3. Organizational requirement: Operational process
    3.1. The system will be used by students only

2.7: Use Case Diagram

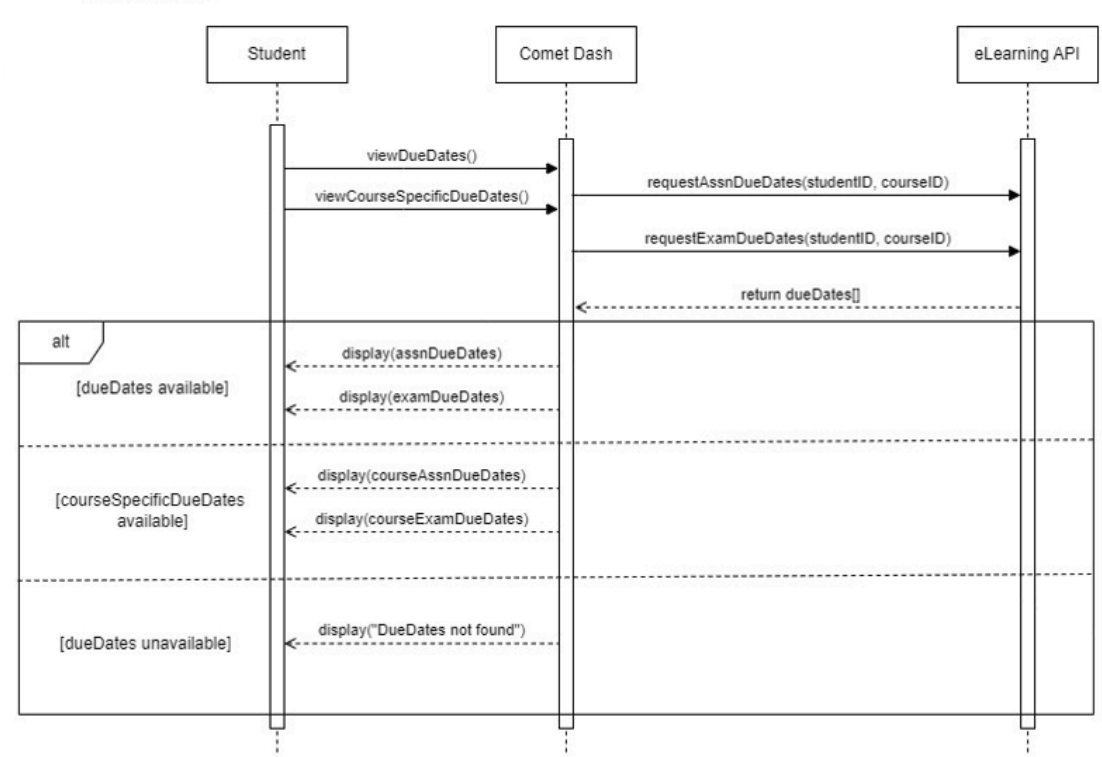


2.8: Sequence Diagrams

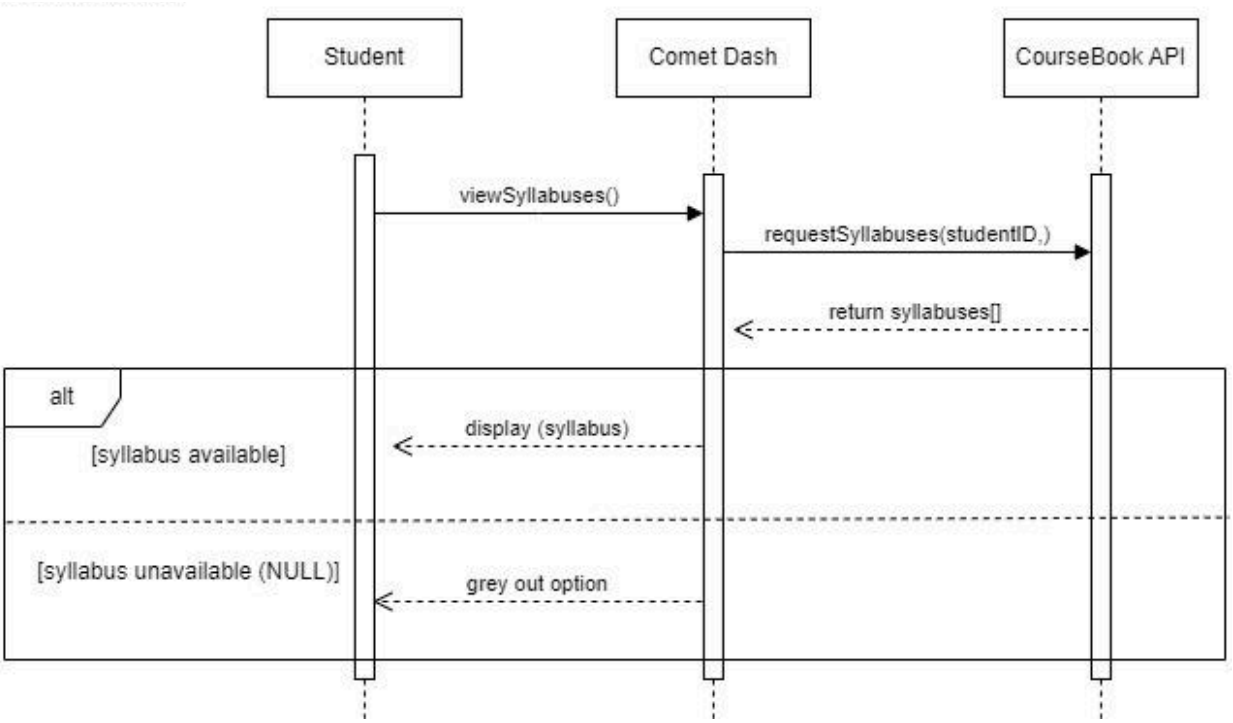
- View Grades



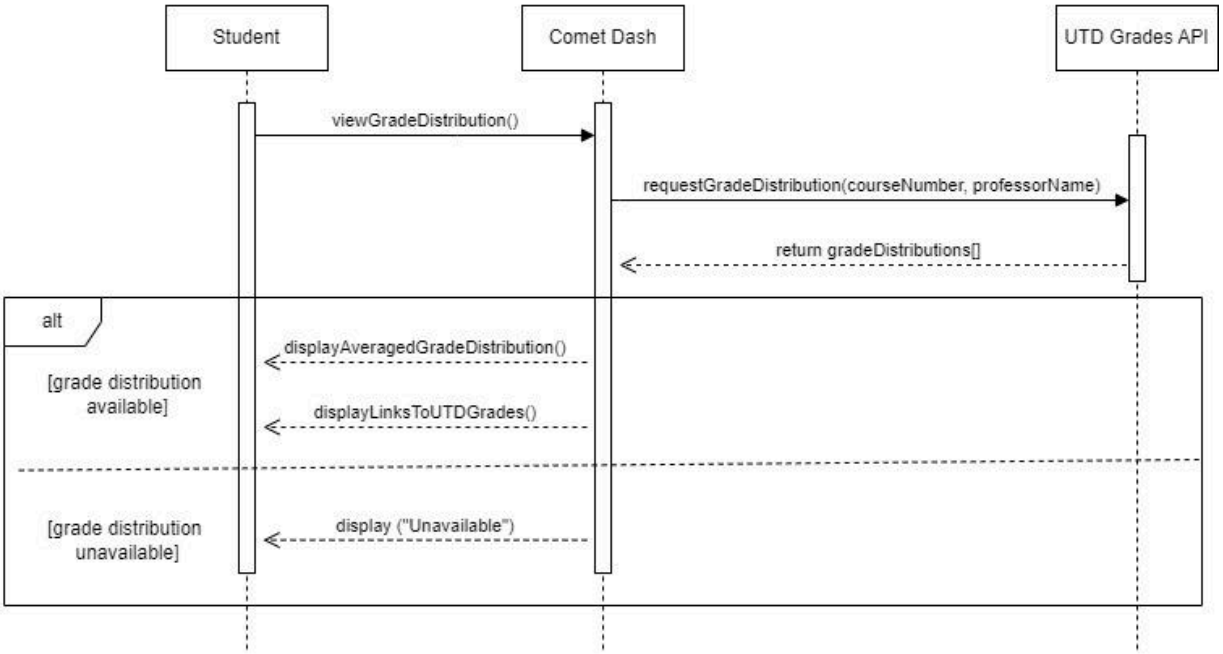
- View Assignment Due Dates



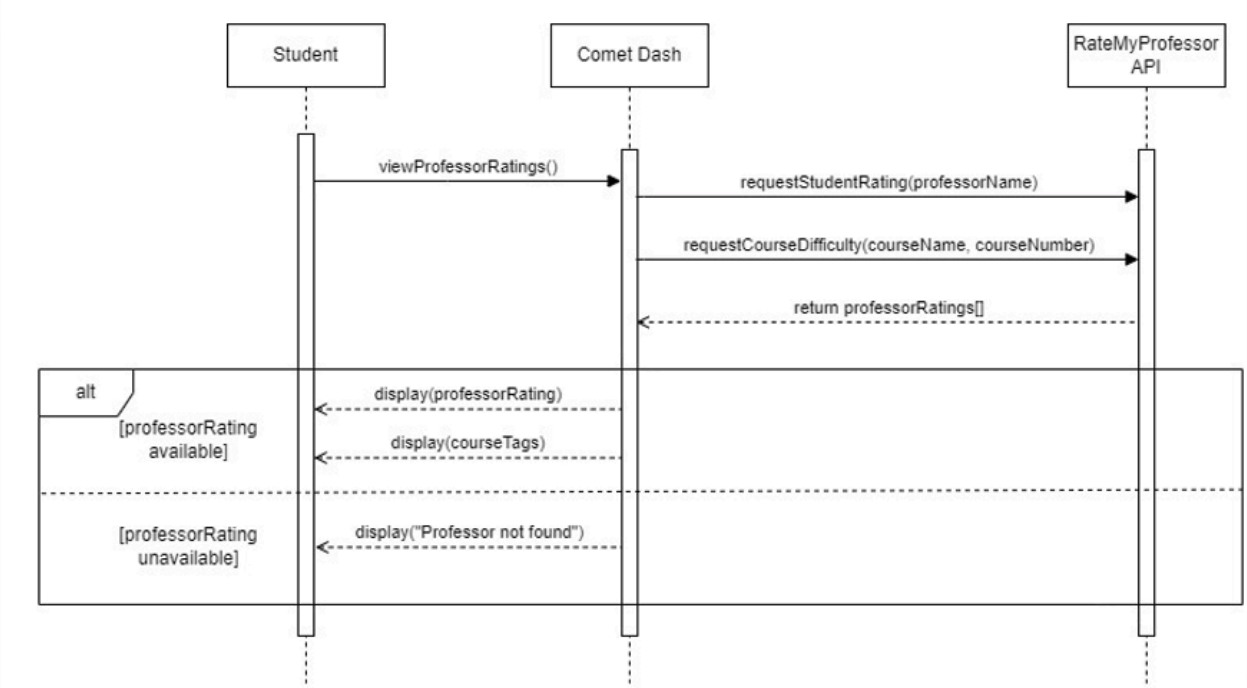
- View Course Syllabi



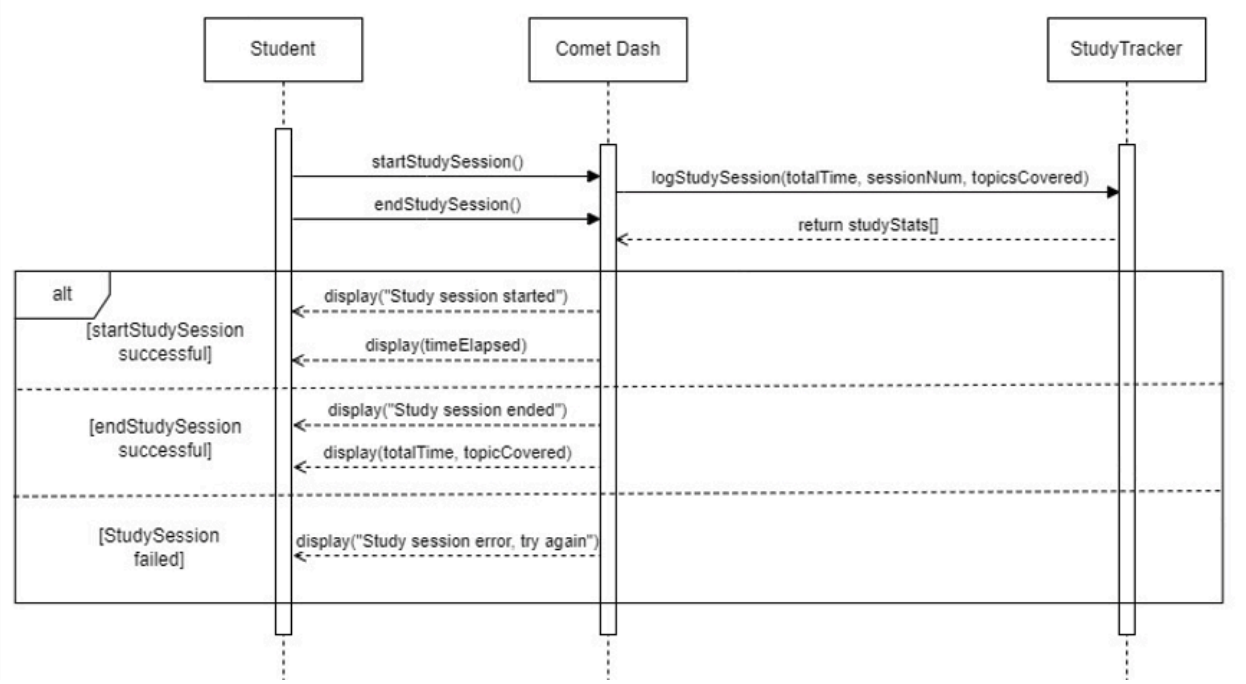
- View Course Grade Distributions



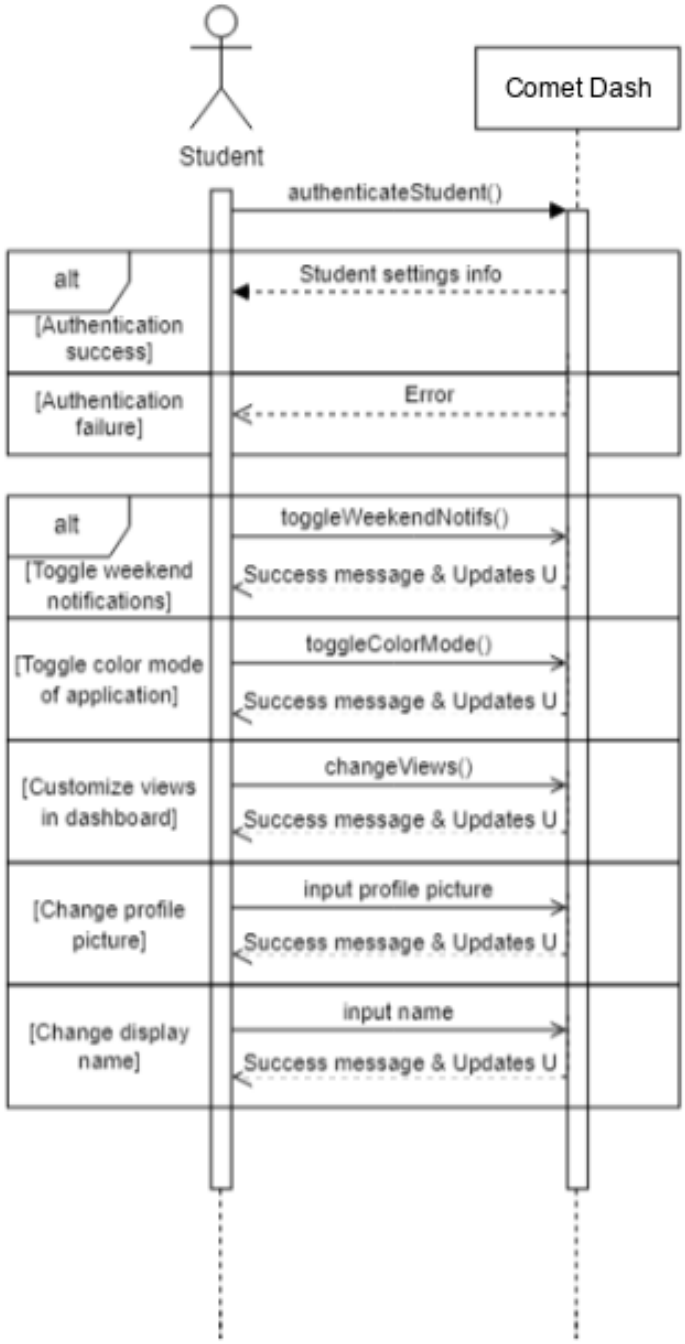
- View Course Rate My Professor Statistics



- Track Study Session

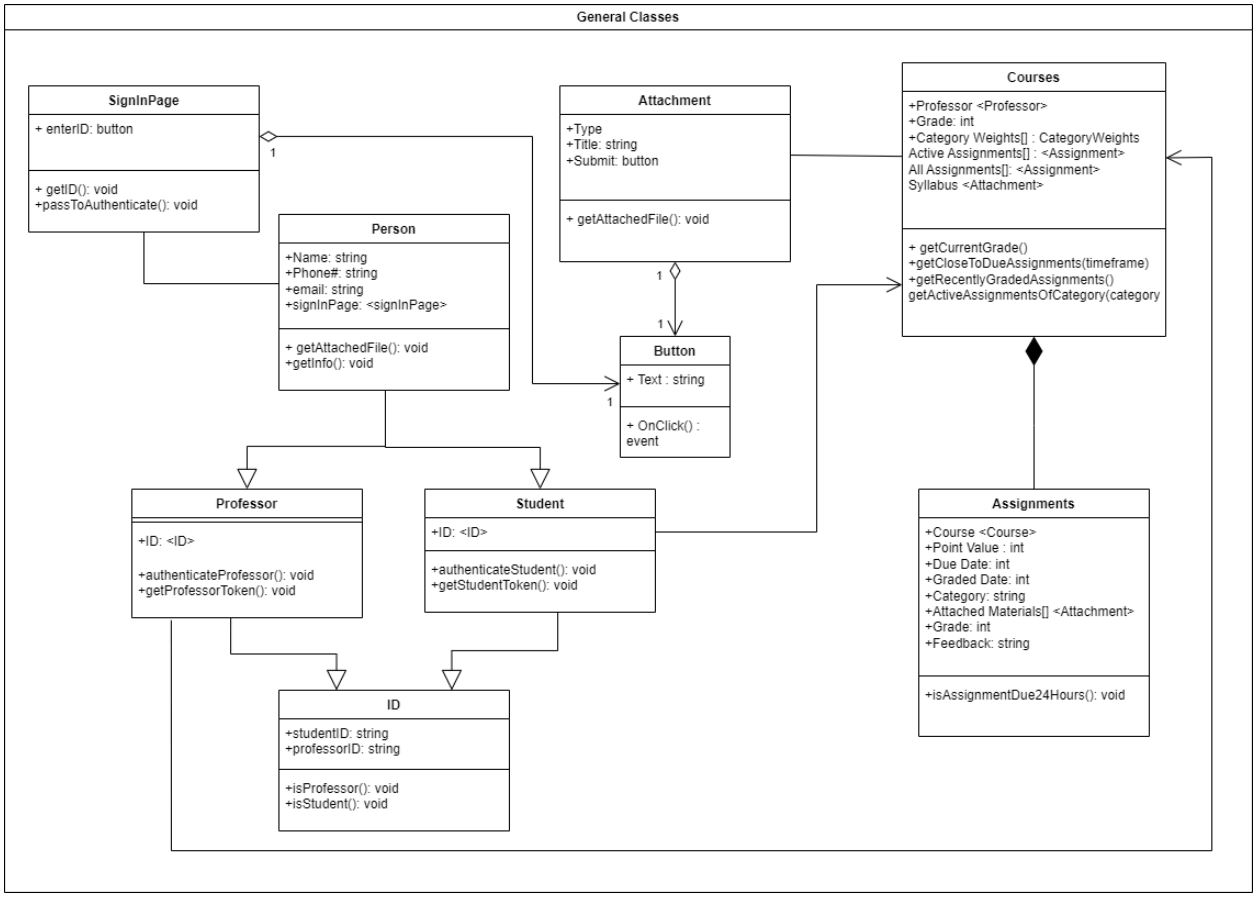


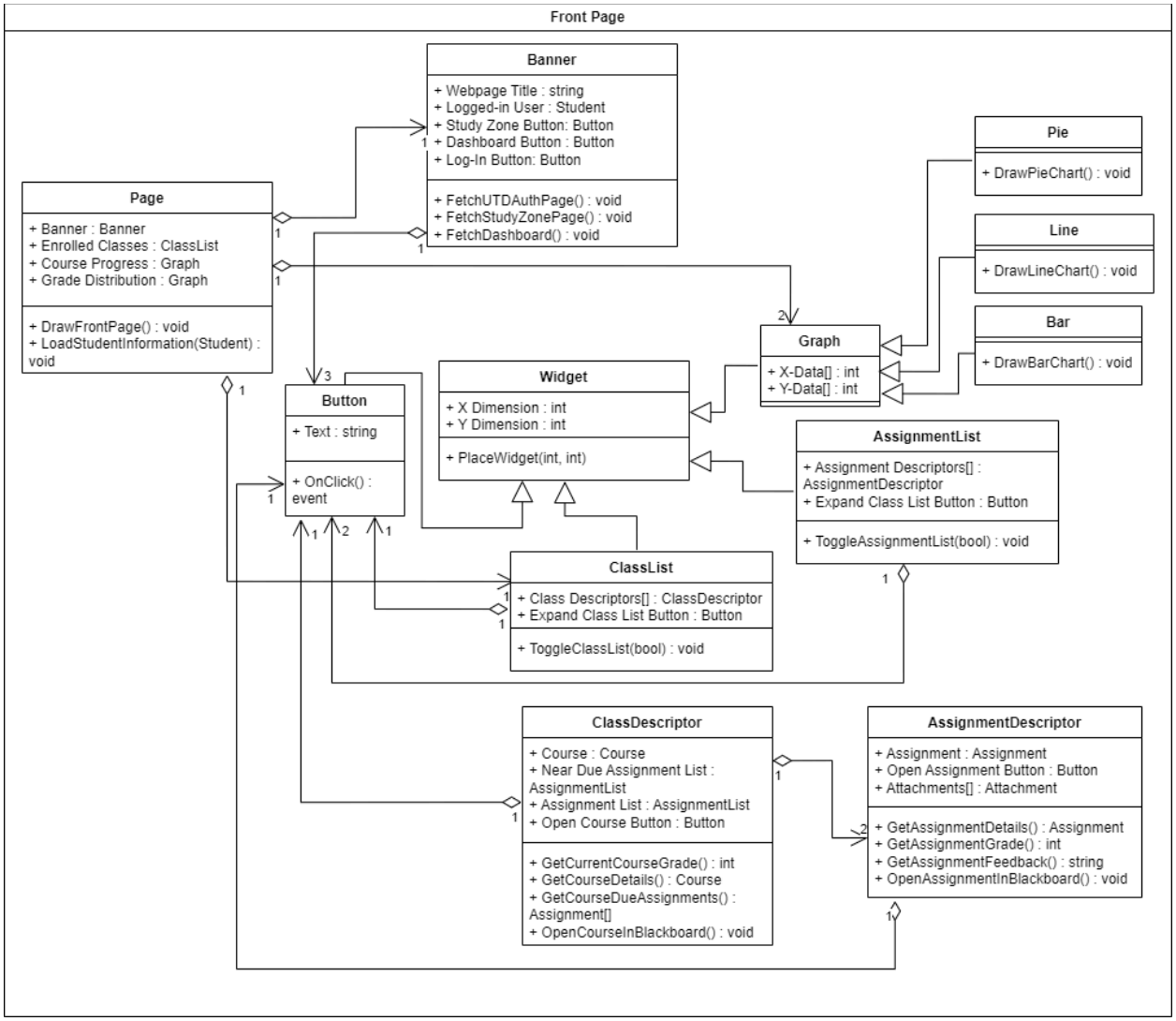
- Modify Settings

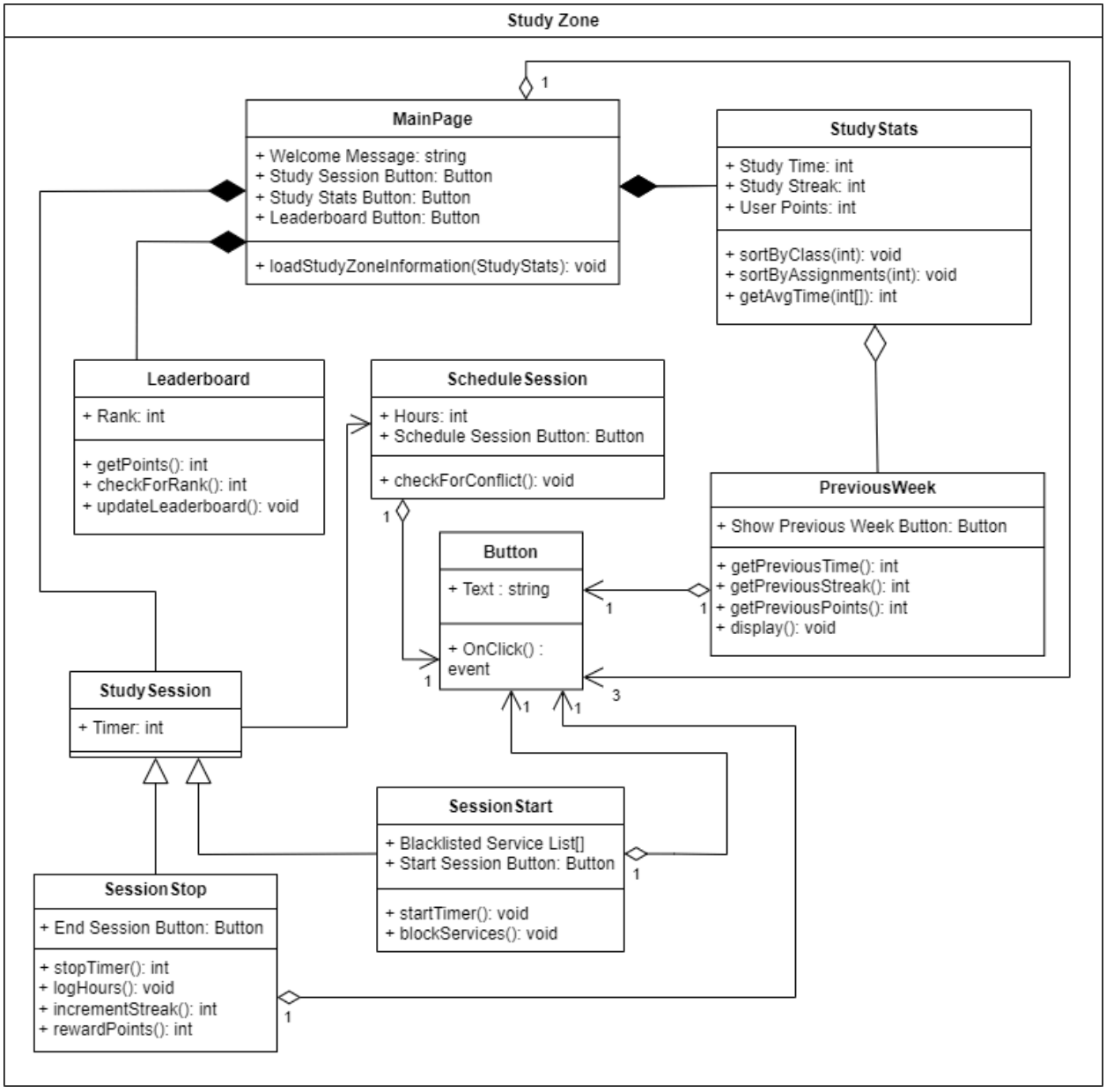


Assumptions:
The API use cases don't have their own sequence diagrams because the student use case sequence diagrams already show how the API functions are used.

2.9: Class Diagrams

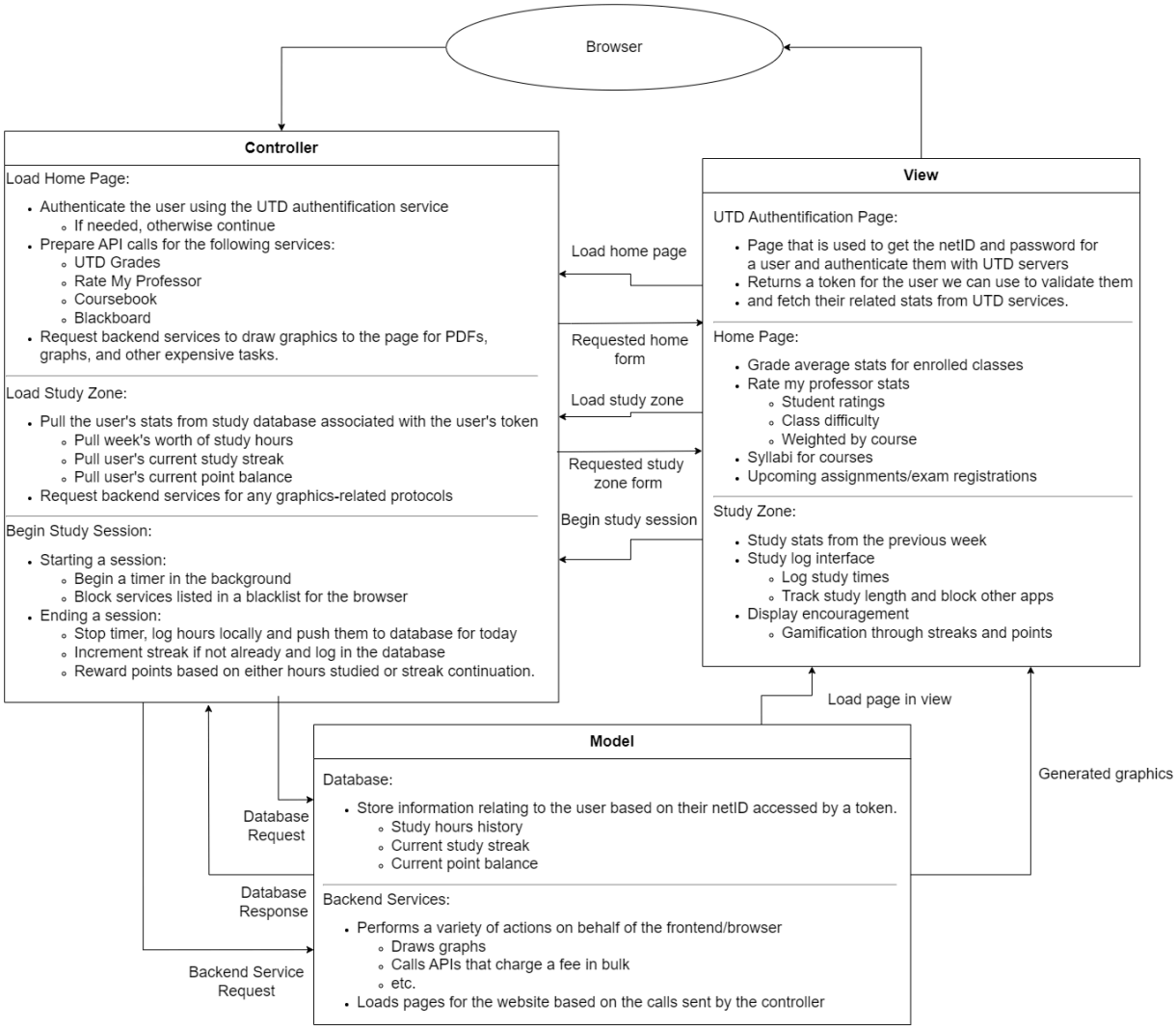



Front Page
Banner
+ Webpage Title : string
+ Logged-in User : Student
+ Study Zone Button: Button
+ Dashboard Button : Button
+ Log-In Button: Button
+ FetchUTDAuthPage() : void
+ FetchStudyZonePage() : void
+ FetchDashboard() : void
Pie
+ DrawPieChart() : void
Page
+ Banner : Banner
+ Enrolled Classes : ClassList
+ Course Progress : Graph
+ Grade Distribution : Graph
+ DrawFrontPage() : void
+ LoadStudentInformation(Student) : void
Line
+ DrawLineChart() : void
Graph
+ X-Data[] : int
+ Y-Data[] : int
Bar
+ DrawBarChart() : void
Widget
+ X Dimension : int
+ Y Dimension : int
+ PlaceWidget(int, int)
Button
+ Text : string
+ OnClick() : event
AssignmentList
+ Assignment Descriptors[] : AssignmentDescriptor
+ Expand Class List Button : Button
+ ToggleAssignmentList(bool) : void
ClassList
+ Class Descriptors[] : ClassDescriptor
+ Expand Class List Button : Button
+ ToggleClassList(bool) : void
ClassDescriptor
+ Course : Course
+ Near Due Assignment List : AssignmentList
+ Assignment List : AssignmentList
+ Open Course Button : Button
+ GetCurrentCourseGrade() : int
+ GetCourseDetails() : Course
+ GetCourseDueAssignments() : Assignment[]
+ OpenCourseInBlackboard() : void
AssignmentDescriptor
+ Assignment : Assignment
+ Open Assignment Button : Button
+ Attachments[] : Attachment
+ GetAssignmentDetails() : Assignment
+ GetAssignmentGrade() : int
+ GetAssignmentFeedback() : string
+ OpenAssignmentInBlackboard() : void


Study Zone
MainPage
+ Welcome Message: string
+ Study Session Button: Button
+ Study Stats Button: Button
+ Leaderboard Button: Button
+ loadStudyZoneInformation(StudyStats): void
Study Stats
+ Study Time: int
+ Study Streak: int
+ User Points: int
+ sortByClass(int): void
+ sortByAssignments(int): void
+ getAvgTime(int[]): int
Leaderboard
+ Rank: int
+ getPoints(): int
+ checkForRank(): int
+ updateLeaderboard(): void
Schedule Session
+ Hours: int
+ Schedule Session Button: Button
+ checkForConflict(): void
Previous Week
+ Show Previous Week Button: Button
+ getPreviousTime(): int
+ getPreviousStreak(): int
+ getPreviousPoints(): int
+ display(): void
Button
+ Text : string
+ OnClick() : event
Study Session
+ Timer: int
Session Start
+ Blacklisted Service List[]
+ Start Session Button: Button
+ startTimer(): void
+ blockServices(): void
Session Stop
+ End Session Button: Button
+ stopTimer(): int
+ logHours(): void
+ incrementStreak(): int
+ rewardPoints(): int
1
1
1
1
1
1
1
1
1
3
1
1

2.10: Architectural Design

MVC Architectural diagram:



3: Project, Scheduling, Cost, Effort, and Pricing Estimation

Project Scheduling Table - Agile Based: Comet DASH

Task	Effort(person-days)	Duration(days)	Dependencies	People on activity
Phase 1: Planning and Requirements			None	Team*
T1: Introductions	4	1		Team*
T2: Requirement Analysis & Assumptions	4	2		Suchita, Sophie
Phase 2: Design and prototyping			T2, T3, T4, T5	Suchita, Sophie, Xander, Bryan, Hassan, Yara, Kimberly, Ronan
T3: Mockup and brainstorm	6	3	T2	Xander, Sophie, Suchita
T4: Wireframe creation	9	3	T3, T2	Suchita, Sophie, Xander
T5: Diagramming and UMLs	20	5	T2	Xander, Hassan Byran, Yara, Kimberly
T6: Prototype Creation	25	5	T5, T4, T2, (M1: approval of final design and prototype)	Hassan, Xander, Yara, Bryan, Ronan
Phase 3: Development			T2, T5, T6, T7(P), T8(P),	Hassan, Xander, Kimberly, Bryan, Ronan, Suchita, Sophie
T8: API Integrations	30	10	T2, T5, T7(P)	Kimberly, Ronan, Bryan

T9: Front end development	60	15	T4, T6, T7, T8	Suchita, Sophie, Yara, Xander
Phase 4: Testing			T2, T5, T6, T7, T8, T9, T10, T11	Kimberly, Bryan, Ronan, Yara
T10: QA Testing	24	6	T6, T7, T8, T9	Kimberly, Bryan, Ronan, Yara
T11: Review of Requirements	8	4	T2, T5, T6(M3: requirement review with stakeholders)	Ronan, Yara
T12: Initial Launch	16	2	T10, T11(M2: initial launch)	Team*
Phase 5: Iteration			T6, T10, T11, T12	Suchita, Sophie, Xander, Bryan, Hassan, Yara, Kimberly, Ronan
T13: Adjustments and Optimizations	150	30	T10, T11	Xander, Hassan, Suchita, Sophie, Ronan
T14: Feedback & Relaunch	40	5	T6, T10, T11, T12(M4: feedback and scrum)	Suchita, Sophie, Xander, Bryan, Hassan, Yara, Kimberly, Ronan

Legend:
- M: Milestone
- T: Task
- P: Parallel task executions
- Team*: Suchita Mamindla, Sophie Chau, Xander Corcoran, Kimberly Niemiec, Yara Abdelhadi, Ronan Rao, Bryan Macias, Hassan Hashemian

<u>Assumptions:</u>

- Workday: 8 hour workday, flexible on time of arrival (assumption that everyone will be working in the same time zone), if not, the team will be local to CST time and adjusted from there. 9:00AM to 5:00PM workdays or alternative 8:00AM to 4:00PM workday.

- Weekend Inclusion: weekends are not required or accounted for in the timeline.
- Agile based process where overall schedule identifies the major phases of the project completion and iterative approach to scheduling.
- Person-days: people on task x day count(days can be split into half and quarter days)

- Project Start Date: Monday, November 11th 2024
- Project End Date: **Wednesday, March 19, 2025**
    - Our project is scheduled to commence on Monday, November 11th , 2024, and conclude on **Wednesday, March 19, 2025** (based on duration).
    - The estimated duration for the project is about 17 weeks, calculated based on a 8 -person team working 8-hour days from Monday through Friday.
    - Weekends will be excluded from the schedule, because it is important to have downtime to brainstorm and think of more efficient solutions to problems. It benefits the team when there is solid downtime and regenerates the team so we do not experience burnout.
    - This results in a total of 84 working days, providing ample time for the team to complete the project's tasks.

      128 calendar days – 44 days skipped: [8]

      Excluded 18 Saturdays

      Excluded 18 Sundays

      Excluded 8 holidays:

        - Thanksgiving Day (Thursday, November 28, 2024)
        - Day After Thanksgiving (Friday, November 29, 2024)
        - Christmas Eve (Tuesday, December 24, 2024)
        - Christmas Day (Wednesday, December 25, 2024)
        - Day After Christmas Day (Thursday, December 26, 2024)
        - New Year's Day (Wednesday, January 1, 2025)
        - Martin Luther King Jr. Day (Monday, January 20, 2025)
        - Presidents' Day (Monday, February 17, 2025)
    - This timeline is made under the assumptions that all team members can work everyday and the agile based approach will not define restrictive deadlines at every point.
        - There will be a 40% contingency plan for problems that may be encountered in the project implementation. People who have other

obligations and must take leave or half-work days will be provided as needed however the plan is flexible with agile development and will be more open towards the working schedule.

Function Point (FP) Estimation:

i. Number of user input
    1. Settings
    2. Study time allocation

ii. Number of user output
    1. Grade distribution report
    2. Study time summary report
    3. Upcoming assignments display
    4. Grades display
    5. Syllabus display
    6. RMP ratings display

iii. Number of user queries
    1. View grades by course
    2. View upcoming assignments by due date
    3. View upcoming assignments by course
    4. Access study time logs by course
    5. Access study time logs by assignment
    6. View grade distribution by course
    7. View syllabus by course

iv. Number of data files and relational tables
    1. User profile data
    2. Study session data
    3. Notification preferences

v. Number of external interfaces
    1. Blackboard API
    2. UTD Grades API
    3. Rate My Professor API
    4. CoursebookAPI

	Function Category	Count	Complexity			Count × Complexity
			Simple	Average	Complex	
1	Number of user input	2	3	4	6	6
2	Number of user output	6	4	5	7	30
3	Number of user queries	7	3	4	6	28
4	Number of data files and relational tables	3	7	10	15	30
5	Number of external interfaces	4	5	7	10	28
					GFP	122

Processing Complexity (PC):

1. Does the system require reliable backup and recovery? **Score: 1**
2. Are data communications required? **Score: 2**
3. Are there distributed processing functions? **Score: 0**
4. Is performance critical? **Score: 1**
5. Will the system run in an existing, heavily utilized operational environment? **Score: 0**
6. Does the system require online data entry? **Score: 2**
7. Does the online data entry require the input transaction to be built over multiple screens or operations? **Score: 0**
8. Are the master files updated online? **Score: 0**
9. Are the inputs, outputs, files, or inquiries complex? **Score: 2**
10. Is the internal processing complex? **Score: 1**
11. Is the code designed to be reusable? **Score: 2**
12. Are conversion and installation included in the design? **Score: 0**
13. Is the system designed for multiple installations in different organizations? **Score: 3**
14. Is the application designed to facilitate change and ease of use by the user? **Score: 3**

Total PC: 17

Processing complexity adjustment (PCA):

PCA = 0.65 + 0.01(Total PC) = 0.65 + 0.01(17) = **0.82**

Function Point (FP):

FP = GFP x PCA = 122 x 0.82 = **100.04 FP**

Cost of hardware products:

- Development workstations: We would need workstations for each developer contributing to the project if they are not expected to provide any. Considering a small team of **8 developers** where we would spend between $1,000-$2,000 per machine, we would have a total expenditure of **$8,000 to $16,000**
- Testing devices: We might invest in a few devices to test the responsiveness of our web app and general compatibility across different screen sizes
    - We would only need to invest heavily in a few devices, a low-end laptop, tablet, and smartphone should suffice. Expected expenditure of **$1,500 to $3,000**

- Servers for development/deployment: To have a strong cloud server to host the website, we would want to invest in a cloud host provider such as AWS, which for our volume of consumer base would cost **$100/month**, totaling **~$1,200/year**.
    - If we host off of the UTD servers, we can greatly minimize the cost depending on the availability of servers and the academic staff's support.

Estimated Hardware Cost: $10,000 - $20,000

Cost of software products:

- Frontend Frameworks and Libraries: We plan to be using open-source library frameworks such as React, Angular, or Vue.js. All of which are free to use.
- Backend server and Database: Cloud-based databases such as AWS or Firebase can cost between **$25-100/month**, leading to **~$1,200/year**.
- Graphing and data visualization libraries: Backend graph and data visualization libraries such as Chart.js, pandas, etc. are open-source and are thus typically free to use.
- API and integration costs: API access to RateMyProfessor is free through open-source libraries and UTDGrades is implemented as an open-source website that we can communicate with.
    - With UTDGrades accessible through its open-source GitHub, we can extrapolate to being able to access Coursebook information for free.
    - Access to data held within Blackboard would have to go through the authority of UTD, either provided for free on a good-faith for-education basis or a premium.

Estimated Software Cost: $500-$1,500

Cost of personnel:

- We are estimating the project to be about a semester-long (15-20 weeks) for a team of 8 people, We expect each person to spend 30-40 hours/wk.
    - When paid, the typical rate for interns or junior developers is about **$20/hr**.
    - At 15 weeks, with 30-40hrs/wk, we expect the cost per person to be **$9,000-12,000**
    - Over the entire team, this extends to **$72,000-$96,000**
    - For an explicitly experienced team, this can extend costs by upwards of another **$10,000**
- For our frameworks and libraries, we may need to provide additional training through online workshops, which could cost upwards of **$500** per person, leading to a total maximal cost of **$4,000** for the entire team.

Estimated Personnel Cost: $72,000-$96,000

OVERALL COSTS: $82,500-$106,500

4: Test Plan

Tested Unit : Study Zone

The purpose of this test plan is to ensure that the Study Zone in our Comet Dash application functions as intended for its users, by offering a user-friendly design, quality software performance, and all advertised capabilities. Our team will therefore be using requirements-based testing to test this unit of our application. Our testing will focus on the core functions of the Study Zone, such as retrieving student study data, accurately displaying the leaderboard, scheduling a study session, and starting/ending a study session. The goal of our tests is to confirm that all functions are operating as expected.

Predicted Testing Schedule :

Day 1 - 2: The focus of these days will be to finalize our test plan and test cases, as well as set up a testing environment for the next steps.

Day 3 - 4: The focus of these days will be to test the core functions of the Study Zone (i.e. retrieving student study data, accurately displaying the leaderboard, scheduling a study session, and starting/ending a study session, etc.).

Day 5 - 6: The focus of these days will be to review the results of our tests and resolve any critical errors that were identified in the Study Zone software.

* Adjustments may be made based on specific needs, team input, or results from unit testing

Testing Records :

Subsequent tests and their results will be recorded with the JUnit testing software. JUnit's testing capabilities allow for clear output, offering easy, efficient identification of whether a test passes or fails based on some predetermined requirement. An example of a typical test with its resulting test cases are attached below. This example serves as a reference for the type of tests we will be running through the JUnit testing software. The results will be recorded and reviewed by the development team for further corrections.

Sample Test and Test Cases :

If a student starts a study session the applications listed on the 'Blacklisted Service List' will be blocked until the session is concluded.

1. Start a study session with no applications in the ‘Blacklisted Service List’. Check that the student is allowed to access any service during the session.

2. Start a study session with one application in the ‘Blacklisted Service List’. Check that the student is not allowed to access that service during the session.

3. Start a study session with two or more applications in the ‘Blacklisted Service List’. Check that the student is not allowed to access those services during the session.

5: Comparison of Work

The currently available similar products can be split into three categories: products that are mostly dashboard-focused, products that deal with education and education-related statistics, and products that offer a combination of both of these features.

Dashboard-focused products are modeled around providing information to the user in an easy to understand way. Usually, these types of products are focused on graphs that illustrate relevant information. For example, the stock charts website Investopedia features graphs and information about some of the most relevant stocks and their respective values/value changes [1]. Similarly, Google trends provides line graphs of the popularity of search terms [2]. This information can then be used to extrapolate the interest of certain topics over a time period. For example, the Google trends graph for the word "election" peaks at around midnight on election night, which likely means that many people were interested in the election around that time [3].

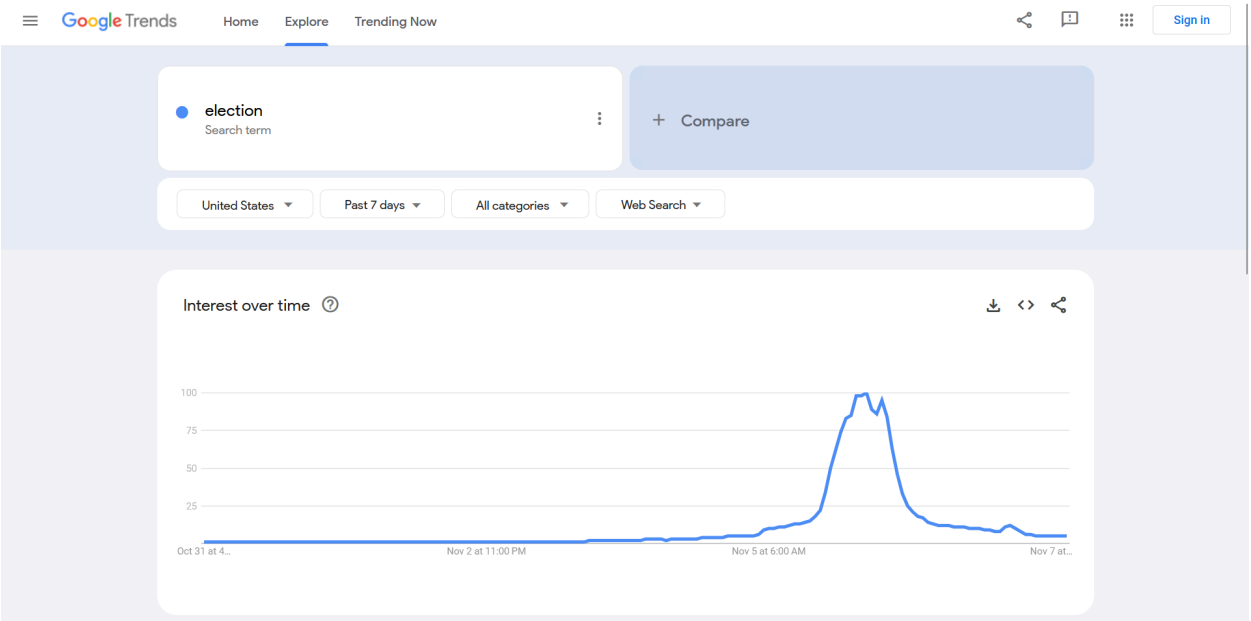


Fig. 1: The Google trends webpage for the search term "election"

More education-focused products generally lack graphs or charts, but still contain information related to education statistics. For example, learning management systems (LMS) are used to keep track of classes, class enrollments, and grades [4]. A popular LMS is the Blackboard system (which at UTD is called eLearning). One of Blackboard's main features is the "Grades" tab, where students can see all of their graded assignments at a glance, including their overall course grade for every class [5].

There are also various websites which provide a mix of the two services. For example, UTD students at Nebula Labs developed a website called UTD grades that shows the grade

distributions for every class taught at UTD [6]. Once users search for a particular class or professor, they are able to easily see a bar chart of grade distributions for the classes they have searched for. This website therefore provides both an insight into students' education and a dashboard-focused design.

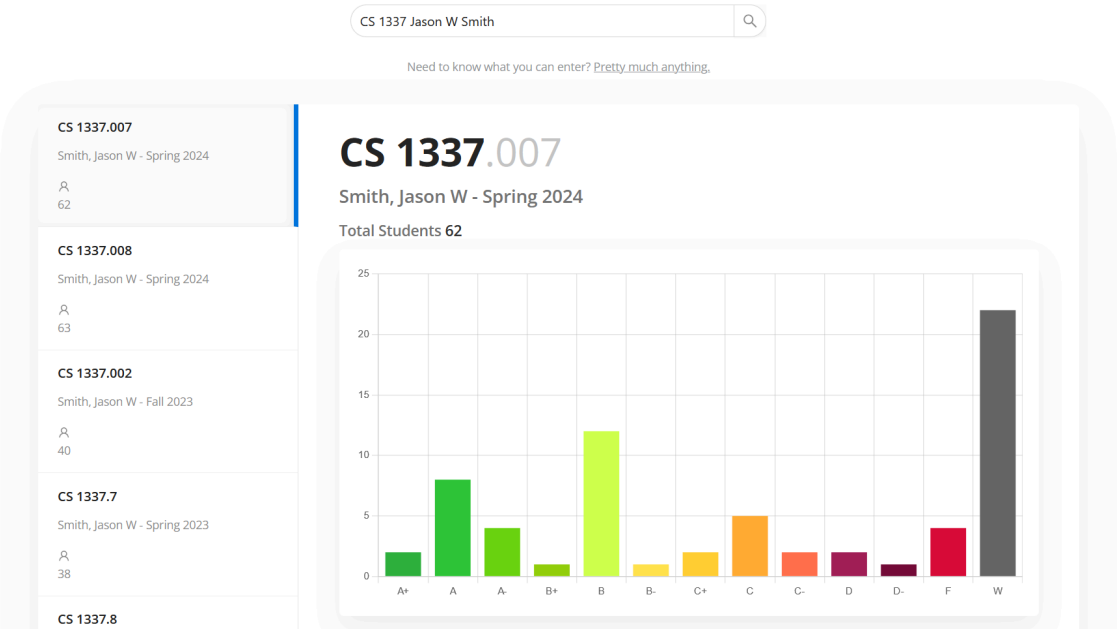


Fig. 2: The UTD grades page when searching for CS 1337 sections taught by Jason Smith [7]

6: Conclusion

The Comet Dash project provides a consolidated solution to help UT Dallas students organize their academic lives, especially beneficial for incoming freshmen. Using API integrations and an Agile methodology, the project aims to unify access to class schedules, grades, and study planning in a user-friendly and visually appealing dashboard. Through an exhaustive and comprehensive development plan that considers points of contingency, we have developed a scalable software development plan that meets the requirements of our project to have a tangible impact for UTD students.

Challenges encountered include balancing the comprehensive design with practicality, as well as acquainting team members with varied levels of familiarity with the tools and methodologies. In balancing the feasibility of our design, we made minor architectural changes in the layout of different pages to provide a more streamlined experience for users. Furthermore, cost and scheduling complexities emerged while we adjusted different sections to the plans outlined between team members. Thorough discussion between team members helped provide a consistent understanding of cost and scheduling that aligns with the project's requirements and timeline. Adjustments to the architecture style and scheduling promoted flexibility and creativity in our design that addresses resource constraints and ensures compatibility with UTD's system. These refinements, along with the allocation of contingency time, have solidified a plan that's realistic and scalable, supporting a positive user experience.

With a thorough project scheduling, cost, and effort estimation, the proposed system provides a beneficial resource for the UT Dallas student body, promoting engagement and academic success.

7: References

[1] "Live stocks chart." *Investopedia*. Available: https://www.investing.com/charts/stocks-charts. (Accessed November 11, 2024).

[2] "Google trends." *Google*. Available: https://trends.google.com/trends/. (Accessed November 11, 2024).

[3] "Google trends: election." *Google*. Available: https://trends.google.com/trends/explore?date=now%207-d&geo=US&q=election&hl=en-US. (Accessed November 11, 2024).

[4] A. Uzialko. "What is an LMS (learning management system)?" *Business News Daily*. Available: https://www.businessnewsdaily.com/4772-learning-management-system.html. (Accessed November 11, 2024).

[5] "Blackboard." *Anthology*. Available: https://www.anthology.com/products/teaching-and-learning/learning-effectiveness/blackboard. (Accessed November 11, 2024).

[6] "UTD grades." *UTD Grades*. Available: https://utdgrades.com/. (Accessed: November 11, 2024).

[7] "CS 1337 Jason W Smith." *UTD Grades*. Available: https://utdgrades.com/results?search=CS+1337+Jason+W+Smith&sectionId=37578. (Accessed November 11, 2024).

[8] "Business Date Calculator: Add/Subtract Workdays, Holidays or Weekends," *Timeanddate.com*. Available: https://www.timeanddate.com/date/weekdayadd.html?d1=11&m1=11&y1=2024&. (Accessed November 11, 2024).

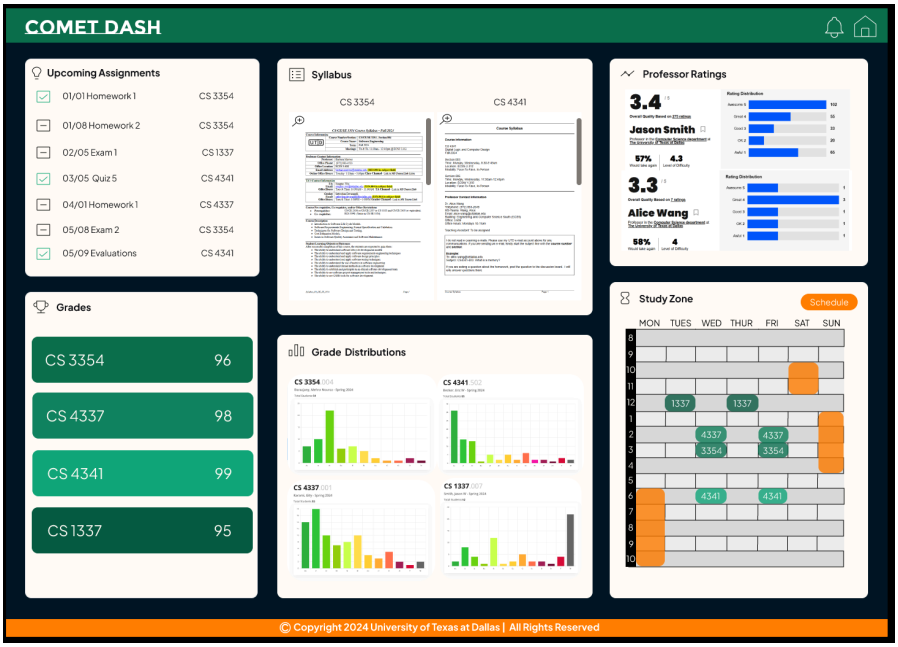
8: Presentation Slides

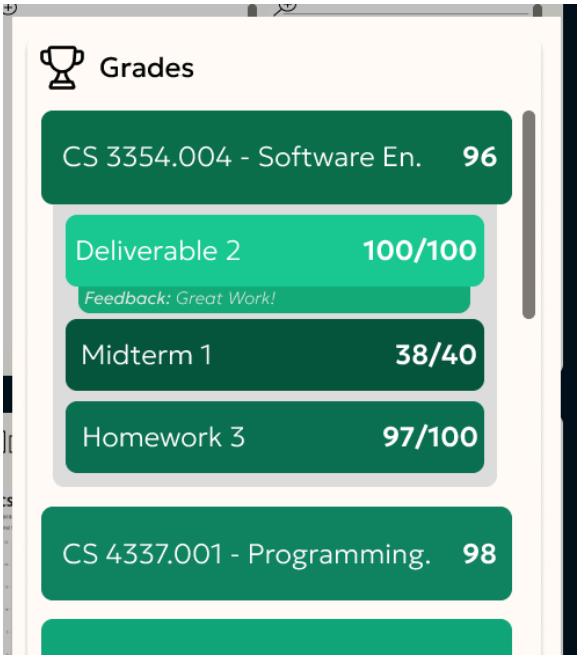
Link:
https://docs.google.com/presentation/d/1CDem-3mBsm62kSx8xu5j65BJz4z1OkqfWC-gWSV4bKk/edit?usp=sharing

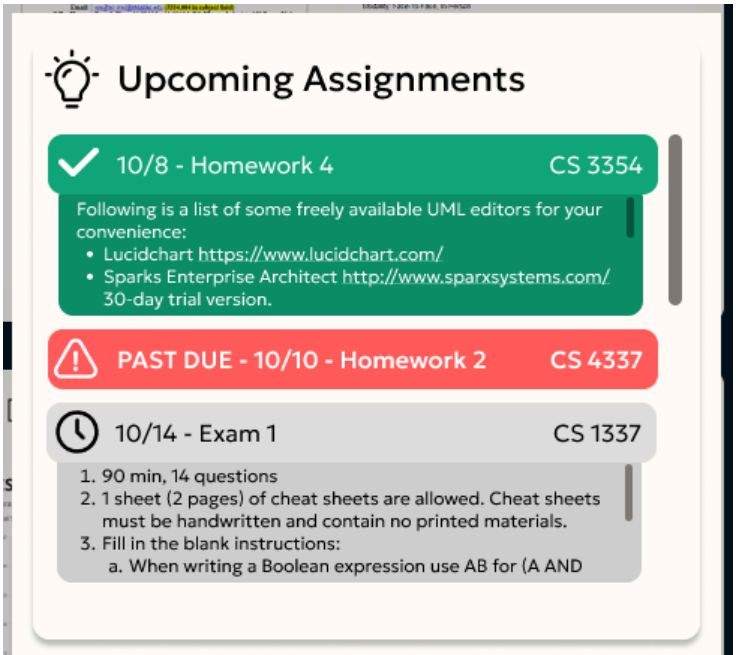
9: Figma Wireframes and Demo

Link:
https://www.figma.com/proto/G4kI0xkTbo6Tq5l742TAaa/Demo?page-id=0%3A1&node-id=1-70&node-type=canvas&viewport=-126%2C395%2C0.35&t=ICwn4FPNkb2V9LcA-1&scaling=scale-down&content-scaling=fixed&starting-point-node-id=1%3A70

Password: CometDash







10: GitHub

Link: https://github.com/IXtimes/3345-softwareEngineeringBaddies