eAttendance

A tablet-based attendance system

Team RISKK

Kalil Garrett

Seth Tomy

Roger Williams

Isaac Avila

Kevin Le

Software Engineering

Spring 2019

Georgia State University

Planning and Scheduling

Assignee Name	Email @student .gsu.edu	Tasks	Duration	Dependency	Due Date	Notes
Seth Tomy	stomy1	• Lead implementation of backend logic • Revise Class diagram	4 hours	n/a	02/02	
Kevin Le	kle24	• Testing • Revise user requirements	4 hours	n/a	02/02	
Kalil Garrett	kgarrett10	• Behavioral modeling • Revise requirement specifications • Assist with frontend/UI • Video	2 hours	n/a	02/02	
Roger Williams (coordinator)	rwilliams179	• Planning & Scheduling • Report • Lead implementation of frontend/UI • Implementation of database design	6 hours	n/a	02/02	
Isaac Avila	iavila1	• Architecture modeling • Revise use case diagram • Assist with backend logic	1 hour	n/a	02/02	

Table 1:Planning and scheduling

Teamwork Basics Summary

Ground Rules

Work Norms

Work norms describe the delegation and execution of activities needed to complete the given tasks. These norms focus on how the team should operate: responsibility distribution, setting deadlines, missed deadlines, quality evaluation, and timeliness.

Example 1: In my negative group experience, work norms were never established, and this led to us being an ineffective team. The only deadline that was enforced was the final deadline, which resulted in certain aspects of the project not being finished and dissatisfaction amongst the group. It would have helped if the group established how to go about handling teammates who miss their deadlines.

Example 2: For another project, a grad student set the deadlines, distributes the work, reviews the work, and deals with group issues since he had a lot of group experience.

Facilitator Norms

Facilitator norms describe the potential use of someone who will drive the project. These norms focus on group progression and problem-solving: deciding whether and how to select a facilitator, rotating and defining the facilitator position, and ensuring project progression.

Example 1: During a hackathon, I became the facilitator of the group. My responsibilities were the following: making sure the team was on the same page, tracking progress of short term and long term goals, and sustaining satisfaction of the group. The group made significant progress on the project and all team members were satisfied at the end of the hackathon.

Example 2: In my experience it's of paramount importance to have a 'facilitator'. When things are left as a group responsibility, things can get forgotten or left out entirely. Thus it is important for someone to take charge/ownership of the group/product.

Communication Norms

Communication norms dictate how and when the group communicates with one another. The medium and time frame should be established early in the group, and it may help to decide on an agreed upon response time (e.g. within 24 hours).

Example 1: I am the president of a new organization called STEMulate, and I work closely with each of the executive members in their role. At the start of the semester, I asked them what is the best way to contact them (e.g. email, phone), and it's been helpful knowing how to contact each of them individually.

Example 2: For most of my previous projects in other classes, we mainly used GroupMe and occasionally do face-to-face meetings after class for more serious matters.

Meeting Norms

Meeting norms describe logistics behind planning meetings. This norm includes the following: knowing individual schedules, responsibility of coordinating or planning meetings, location, and handling tardiness and absences.

Example 1: In a previous class, my group planned meetings as needed since the task at hand did not require us to meet on a set schedule. However, most of my other group experiences involved having a set meeting time. Handling tardiness and absences is not typically addressed, and I think it would help with accountability in the future.

Example 2: There was no actual coordination for meeting schedules for one of my other group projects since we had in-class time for this. However, there were times when one member set up a few meetings when we're trying to reach a major milestone.

Consideration Norms

Consideration norms describe our group environment. This norm focuses on creating a comfortable, productive environment for everyone: is there eating, drinking, smoking, or conversation dominators?

Example 1: Specifically in the past, when meetings have been long(i.e. longer than an hour), everyone's comfort is important for continued interaction. Allowing snacks, coffee, smoke breaks, etc. were helpful in achieving this.

Example 2: Another important factor is meeting time. When meeting times are established by the facilitator without consideration of the team, meetings are missed or there is dissatisfaction among the team on meetings before they even begin.

Hints for Handling Difficult Behavior

Overly Talkative

If someone is dominating the discussion, try to direct the conversation to someone else when there's a pause. If that person remains very talkative, one other person should consult them privately about sharing speaking time.

Example 1: Picking holes in the conversation can be helpful, as well as directing the conversation towards an area that is not that persons expertise. If all else fails a 1-on-1

Example 2: A simple phrase such as, "We know X has all the answers, now would anyone else like to participate," has a two-fold effect. Person X will tend to talk less, and the rest of the group will tend to pitch in.

Too Quiet

Draw out unengaged or unconfident people by asking them about opinions or stuff about themselves. If they're unsure of themselves, compliment them from time to time when they do good work.

Example 1: On the executive board of an organization, there was a person who didn't say much at the first meeting, and a few of us noticed. At future meetings, we made sure to ask them about their opinion, how their tasks are going, and how they're doing as a person.

Example 2: I tend to not talk much because I'm unsure of my knowledge and skills. Another group member for another project kept pushing for my feedback and I told them feedback on what we're doing, which was appreciated and taken into consideration.

Argues

It can be beneficial to have someone that questions the group's idea since it can lead to improvements. However, someone who judges others can destroy the morale of the group. Another member should let them know that their actions have not only negative a negative impact on the individual but the entire team.

Example 1: I have not had experience with a person who is critical of others. This person may be critical of others when the other person is not in the conversation, but this behavior should be addressed by any group member that hears it. I myself may be critical of the group's ideas, and I ask questions to clarify any confusion about the idea.

Example 2: When something feels off, I sometimes speak up about this feeling. For the most part, what I say is good feedback for the team.

Complains

If the person's complaint is legit, set some time to solve it. If not, then help that person improve with what their problem is.

Example 1: In a previous group, two members had not heard of the algorithm we were to implement. I suggested that they do some research to learn more about the algorithm and then think about how they can apply it to our specific application. Unfortunately, they didn't take these steps until the end of the project and continued to complain until they understood the algorithm.

Example 2: I complained about not knowing about database stuff but was pushed to doing it since nobody else in the group knew about databases and the others simply said to try anyway. I still ended up doing the database stuff (kind of) and did well enough to get the project functional.

Hints for Handling Difficult Behavior

Floundering

Floundering is equivalent to prolonging the start of the project, and this happens at the beginning of the project when the team isn't sure what to do. This can be fixed by writing down the tasks that need to be completed.

Example 1: My last group floundered at the start by not knowing how to tackle our problem. We distributed larger responsibilities amongst one another, but none of us knew where to begin. Hence, it took a while for any progress to be made.

Example 2: I am generally focused when in meetings, so if there’s too much idle talk or nothing is happening, I usually ask what we are doing, which usually helps people go back to talking about the project.

Going Off on Digressions and Tangents

Slight tangents are fine for getting to know each other, but too much can detriment progress. Try to get the conversation back to the main topic.

Example 1: If I’m familiar with a person, I may discuss non-project related topics, but I do not do this in a group setting. Sometimes, my groups may go a bit off topic, but the leader or facilitator will bring us back to the topic.

Example 2: When the group decided on a project topic, one group member was very ecstatic about the project being about their idea and kept on talking about implementation when it’s very early in the semester.

Making a Decision Too Quickly

Someone who is eager to start working on an aspect of the project may want to make a quick decision. Someone should ask about what is needed to make this decision and what everyone else thinks.

Example 1: I work closely with someone who is action-oriented in my organization STEMulate. They typically want to apply any advice and critiques before I finish giving them all, but we need to decide on how to go about improving what we’re working on. In previous group assignments, the group must select a group topic, and there’s typically a person who has to do a specific topic. However, I make sure everyone’s voice is heard before the topic is selected.

Example 2: A group member for another project wanted to get things done and used certain hardware without consulting everyone in the group, which made us waste time because it ended up creating arguments on which hardware was better to use for the project.

Not Making a Decision

Be open with others’ feedback. Remember that decisions are for the group, not for one person. If you have trouble, you can do multi-voting: voting for things to be considered and then voting between the top choices. You can also do “Plan A”, where each member allocates 100 “points” to each choice and the one with the most points is chosen.

Example 1: In my past, either the leader or a majority vote would break a stalemate. Typically, the pros and cons are not discussed before the overrule from the leader or majority. I like this new approach which helps to facilitate quicker, and fair decisions.

Example 2: A group member from another project always suggest multi-voting every time we are split or unsure on decisions.

Feuding Between Group Members

Feuds generally impede the group's progress. If there is a feud between group members, the parties involved should discuss the problem using various listening techniques discussed in this Teamwork Basics.

Example 1: Many times when there's a been a feud the best course of action was to discuss the issue with that person and find some sort of common ground. The issue at hand is not always solved but typically the feuding members will find commonality and it will help with future issues.

Example 2: There was a case where two people would argue between different methods of doing certain components of a project. I told them to focus on what method would best fit this project instead of having biased preference and we ended up using a completely different method.

I**gnoring or Ridiculing Others**

Due to comfortability, group members may sometimes not interact with one another which can lead to cliques within the group. To combat this, every member should attempt to work with every other member.

Example 1: In my negative experience with groups, other members would read the group chat but not respond. This may be because they did not have any project progress or consciously did not want to interact. Work ethic may have been a divide in my previous groups.

Example 2: I tend to feel like I get left out of some important decisions and in one of my other group projects, someone decided to use certain hardware and dumped that responsibility onto me for being the group's best programmer without consulting me about the hardware being used in advance.

The Group Member Who Does Not Do Their Share of the Work

If a member will not work with others, complete their assigned task, or come to meetings, someone should meet with them and discuss their impact on the group.

Example 1: I try to avoid conflict if both parties are not interested in the topic, and I usually pick up the other person’s responsibilities. I am quick to assume the other party does not care about the project, but it will only help both of us if I can approach them and reach a resolution.

Example 2: I don’t usually go to group meetings since they tend to be too much out of my way (I commute to GSU) so I let others who went tell me the gist of the meetings. Had I gone to some of those meetings, I could have given them vital feedback on a decision that they made too quickly. In cases of group members not doing their share of work, I tend to be the one picking up the slack and then consult that member about it privately.

Problem Statement

The tablet-based attendance system (RISKK) will solve the problem of tracking, tabulating, and calculating students' attendance. The target customers will be colleges and universities and will be primarily used by instructors and students. RISKK will eliminate the hassle of paper-based attendance tracking by allowing each student to sign the attendance register during class time by using a tablet computer. With the attendance data gathered from the students, RISKK will provide attendance reports to instructors, calculate and assign attendance grades to students.

This project is feasible and can be implemented with the currently available resources because instead of creating a new, separate solution or system, RISKK will integrate with the institution's existing learning management systems. While current competing products use invasive and complicated biometric authentication methods like facial recognition, RISKK will differentiate itself from its competitors by using less invasive methods while still ensuring security, privacy, and reducing fraud. The most interesting technical aspects of this product will be how it solves the student authentication and impersonator problems and how it will seamlessly integrate with the various proprietary learning management systems used by various colleges and universities.

Requirements

User Requirements

The software (eAttendance) shall be designed as a secure and ease-of-use tablet-based attendance system that focuses on streamlining the attendance taking process. The software shall utilize the school's learning management system (LMS) API to access data that's already accessible when using the LMS. A database should store data for every class roster, which contains instructor identification to link them to the class and a list of times of when attendance was taken. Each attendance session times contain a list of students with their associate signature (or no signature if the student was absent) to determine whether that student attended. Upon starting the software application, a login page should appear, prompting that login verification is required. Since the system requires verifying and accessing data in a database, an internet connection is required to use this software's features. Once logged in:

- The user is sent to the main menu, where the user will view a list of classes that they are deemed as an instructor (using the school's LMS). Not all classes will necessarily be shown in the list as an instructor could also be a student of another class.
- Instructors shall be able to view the data from all class rosters in the database that have a matching instructor identification by selecting a class roster among the classes they teach in the main menu and clicking a button.
- Instructors shall be able to deem a student's signature as invalid (which change's a student's attendance at that attendance session to absent) for cases where an instructor sees that a signature does not actually verify the student's identity, such as using a straight line as a signature or if a signature looks forged.
- To start taking attendance, instructors would select a class roster among the classes they teach in the main menu and click a button to start the attendance taking process.
- While attendance is being taken, students should only be able to view the class roster by selecting their name in a class roster list and signing in via signature. After they've signed in, the student would briefly see their attendance grade and any attendance warnings generated by the software.
- When an instructor wants to finish taking attendance, they would need to use a passcode verification (the school's LMS verification system may be used) so that their students can't interfere with the attendance system. This should then send all collected data from the attendance session to the database for the instructor to store and view. When that is done, the user is sent back to the main menu.
- If an instructor wants to cancel taking attendance, it would work the same way as finishing the attendance process except that data will not be collected.
- Instructors shall be able to make corrections to their students' attendance data for every class that they're teaching in case of a student-made error, such as a student signing in as another student by accident. They should not be able to add or remove any data. This may also be done while attendance is being taken, but it would require a passcode verification like the one used for when an instructor wants to finish taking attendance.

- Instructors shall be able to log out of the system while logged in, leading the user back to the login page. However, instructors cannot log out while they are taking attendance since students could mess with the attendance process by logging off.

System Requirements

Context Model Diagram

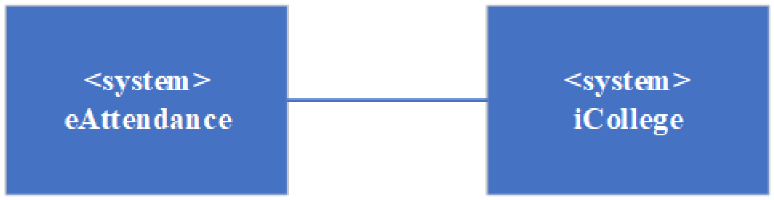


Figure 1: Context model diagram

Use Case Diagram

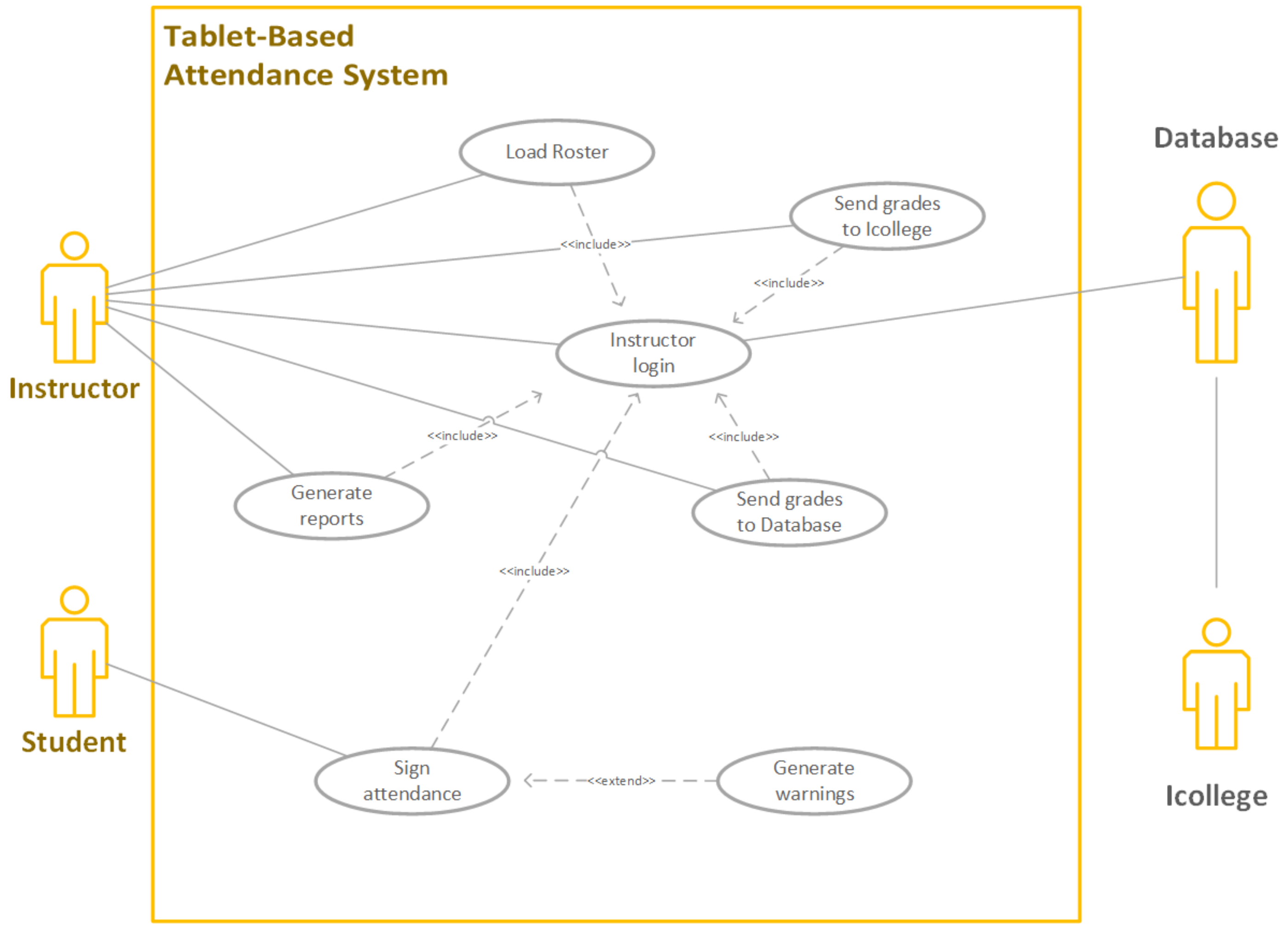


Figure 2: Use case diagram

Requirement Specifications

Summary (Instructor login): The instructor connects the application to their iCollege account.

Basic Course of Events:

1. Open the Tablet-Based Attendance Application
2. The instructor is sent to our login page and asked to signed in.
3. If login is successful, complete use case *Load roster*.
4. Show confirmation message that application has successfully connected to their iCollege account and proceed to next screen.

Alternative Paths: In step 2, the instructor can cancel their login and quit the application.

Exception Paths: In step 2, if the login fails, the instructor will need to repeat step 2 or quit the application. In step 3, if verification fails, we will need to either (a) ask for permission or (b) show an error message.

Precondition: The instructor has a powered tablet that is connected to wifi and the Internet.

Postcondition: The application is successfully connected to iCollege.

Summary (Load roster): The application should be able to access class rosters.

Basic Course of Events:

1. The instructor's token for iCollege will be sent to iCollege to pull the class rosters.
2. If the class rosters are successfully accessed, raise an ok status.

Exception Paths: In step 1, if unsuccessful, raise an error status.

Precondition: The instructor has successfully logged in to iCollege and has been given a token.

Postcondition: The application has successfully downloaded the rosters.

Summary (Start attendance session): The instructor begins a new attendance session for a course.

Basic Course of Events:

1. The instructor is brought to a page where they can enter additional information about the session (e.g. there is a test this class). The time and date are automatically included.
2. The instructor selects the course that this attendance session belongs to.
3. The class roster is pulled from the selected course.
4. An attendance form is generated.

Alternative Paths: In step 1 and 2, the instructor can cancel the creation of the attendance session.

Precondition: Completion of use case *Instructor login*

Postcondition: Students can begin to sign in.

Summary (Sign attendance): The students sign in using the tablet.

Basic Course of Events:

1. The student selects their name from the class roster.
2. The student will give their signature.
3. The student's attendance information will be stored locally on the tablet.
4. If the information is stored on the tablet successfully, the students receive a confirmation message that must check that they've seen.
5. Complete use case *Generate warnings*
6. The student passes it on to the next student.

Exception Paths: In step 2, if the name is not in the roster, students will be sent a message to enter their name as it would appear in iCollege and sent back to step 1. In step 1, if a student has already been signed in, the student can click a button that will notify the instructor that someone has signed in for them.

Precondition: Completion of use case *Start attendance session*

Postcondition: The student's information session has been successfully stored on the tablet.

Summary (Generate warnings): The application creates warnings and displays warnings to students.

Basic Course of Events:

1. The student's attendance history is pulled from the database.
2. Warnings are calculated by the application using either default warnings or custom warnings.
3. The warning is shown to the student for 4 seconds.

Precondition: Completion of use case *Sign attendance*

Postcondition: Students are aware of their current attendance and are shown warnings if necessary.

Summary (Close attendance and view summary): The instructor will sign off that the attendance session is over and receive an attendance summary for that session.

Basic Course of Events:

1. The instructor closes the sign-in session and gives their signature.
2. The information is transferred from the tablet to the database.
3. If the instructor chooses, completion of use case *Generate summary*

Alternative Paths: The instructor can choose to skip step 3.

Exception Paths: In step 2, if the attendance session is not successfully transferred, display an error message.

Precondition: Completion of use case *Sign attendance*

Postcondition: The attendance session is successfully added to the database, and the instructor can view a summary if desired.

Summary (Generate report): The database generates a summary for the given attendance session(s).

Basic Course of Events:

1. The instructor selects which attendance session(s) to include in the summary.
2. The database sends the selected session(s) information back.
3. The application formats the information into a viewable document.

Alternative Paths: In step 1, the instructor can choose to cancel generating a summary.

Precondition: The attendance session(s) have been stored in the database.

Postcondition: A formatted document with information on the selected session(s) is viewable.

Summary (Send grades to iCollege): The instructor can choose to send grades to iCollege.

Basic Course of Events:

1. The instructor clicks on a button to be taken to their courses.
2. The instructor clicks a button to send either all courses' grades or one course's grades to iCollege.
3. The instructor's token and course's attendance information is pulled from the database and sent to iCollege.
4. If successful, show a confirmation message.

Alternative Paths: In step 2, the instructor can go back to the main dashboard. In step 3, the instructor can cancel sending grades to iCollege.

Precondition: Completion of use case *Instructor Login*

Postcondition: Student's attendance grade is updated on iCollege.

Summary (Send grades to database): The instructor can choose to send grades to the database.

Basic Course of Events:

1. The instructor clicks on a button to be taken to their courses.
2. The instructor clicks a button to send either all courses' grades or one course's grades to the database.
3. The attendance information stored on the tablet is sent to the database.
5. If successful, show a confirmation message.

Alternative Paths: In step 2, the instructor can go back to the main dashboard. In step 3, the instructor can cancel sending grades to the database.

Precondition: Completion of use case *Instructor Login*

Postcondition: Student's attendance grade is updated on iCollege.

Summary (Set warnings): The instructor can create custom warning messages to be sent to students after they sign in if they've missed a certain number of classes.

Basic Course of Events:

4. The instructor enters a page to manage warnings.
5. To add a new warning, the instructor asks to add a new warning.
6. The instructor enters the number of classes that must be missed to display a warning.
7. The instructor types in a custom message for when a student misses said number of classes.
8. The instructor selects which courses to apply this warning to.
9. The instructor selects how often this warning will be sent once a student misses said number of classes.
10. The instructor can choose to have warnings be emailed to students.

Alternative Paths: In step 2, the instructor can modify an existing warning and proceed to step 3. In steps 2-6, the instructor can cancel adding or modifying a warning.

Precondition: Completion of use case *Connect to iCollege*

Postcondition: Courses have a warning message that they can show after a certain number of absences.

System Modeling

Class Diagram

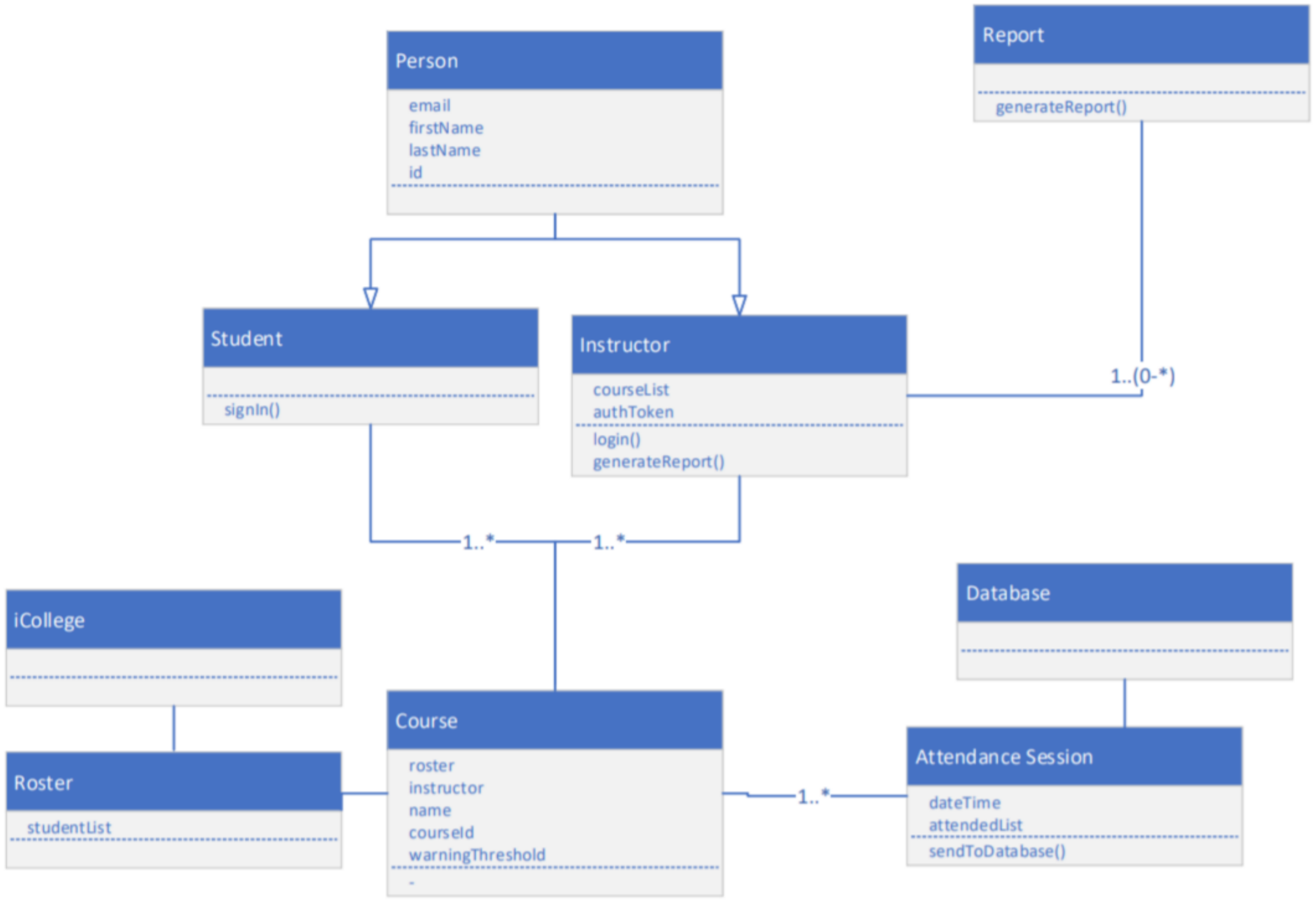


Figure 3: Class diagram

Database Specification

The Tablet-Based Attendance system will use the Firebase database management system (DBMS). The main entities of the database are: Student, Instructor, Course, and Attendance. Figure 3 depicts the database tables, each with their attributes and data types, and the relationship between them.

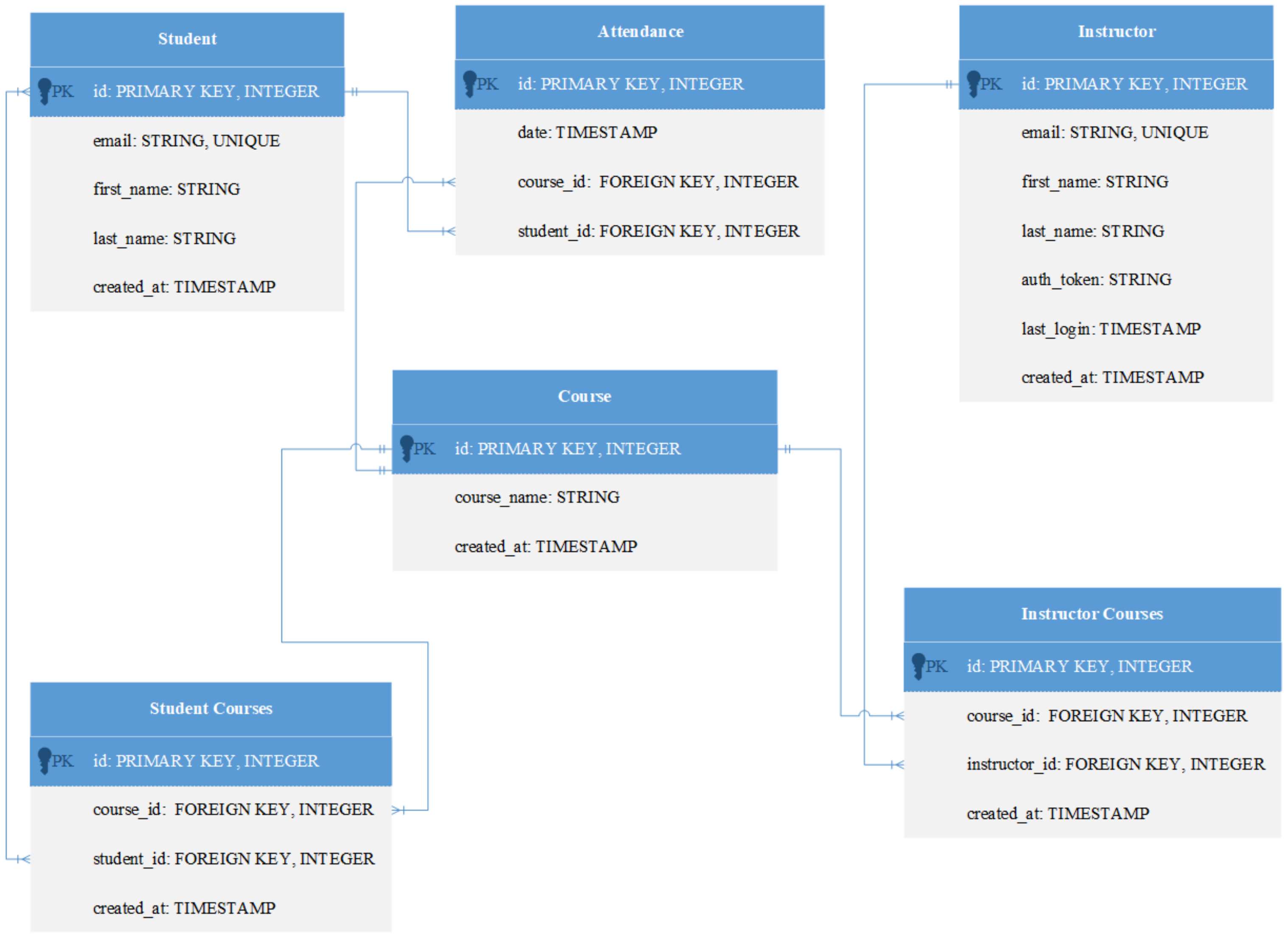


Figure 4: Entity Relationship Diagram

Architecture modeling

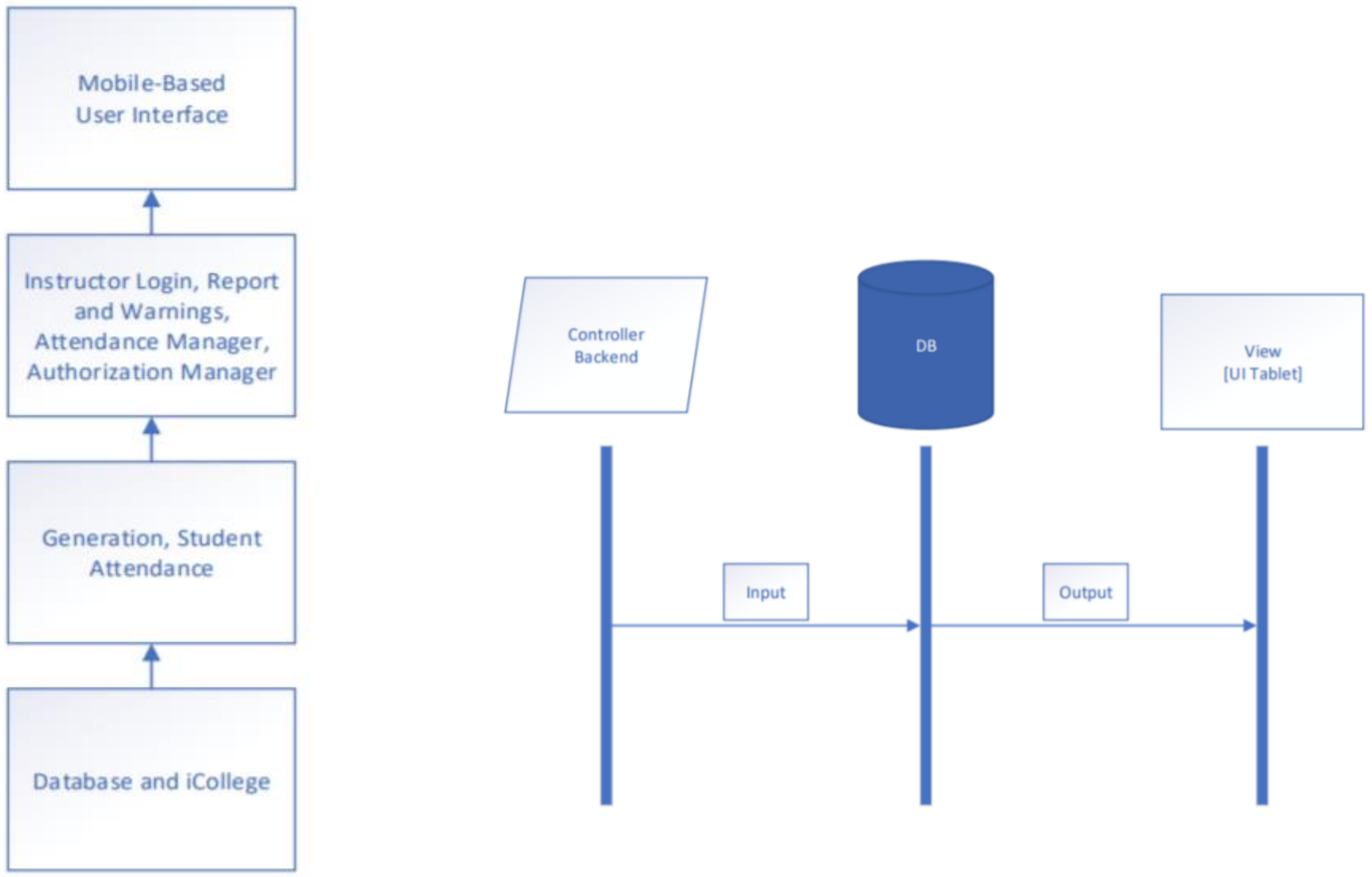

Mobile-Based User Interface
Instructor Login, Report and Warnings, Attendance Manager, Authorization Manager
Generation, Student Attendance
Database and iCollege
Controller Backend
DB
View [UI Tablet]
Input
Output

Figure 5: Layered architechture model and MVC pattern

Behavioral modeling

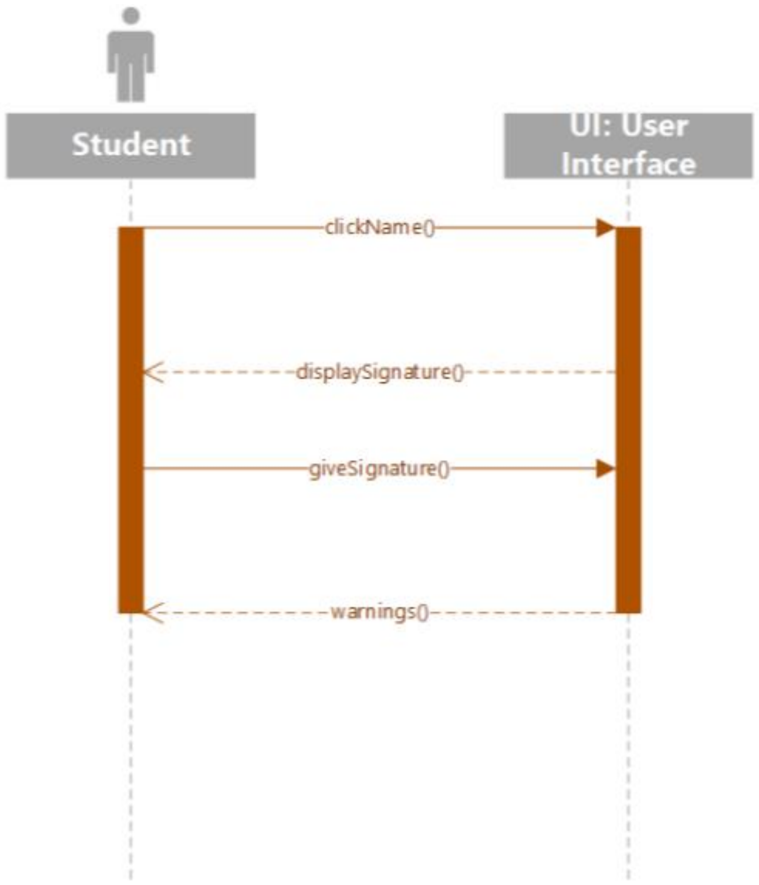


Figure 6:Sequence Diagram for the professor login use case

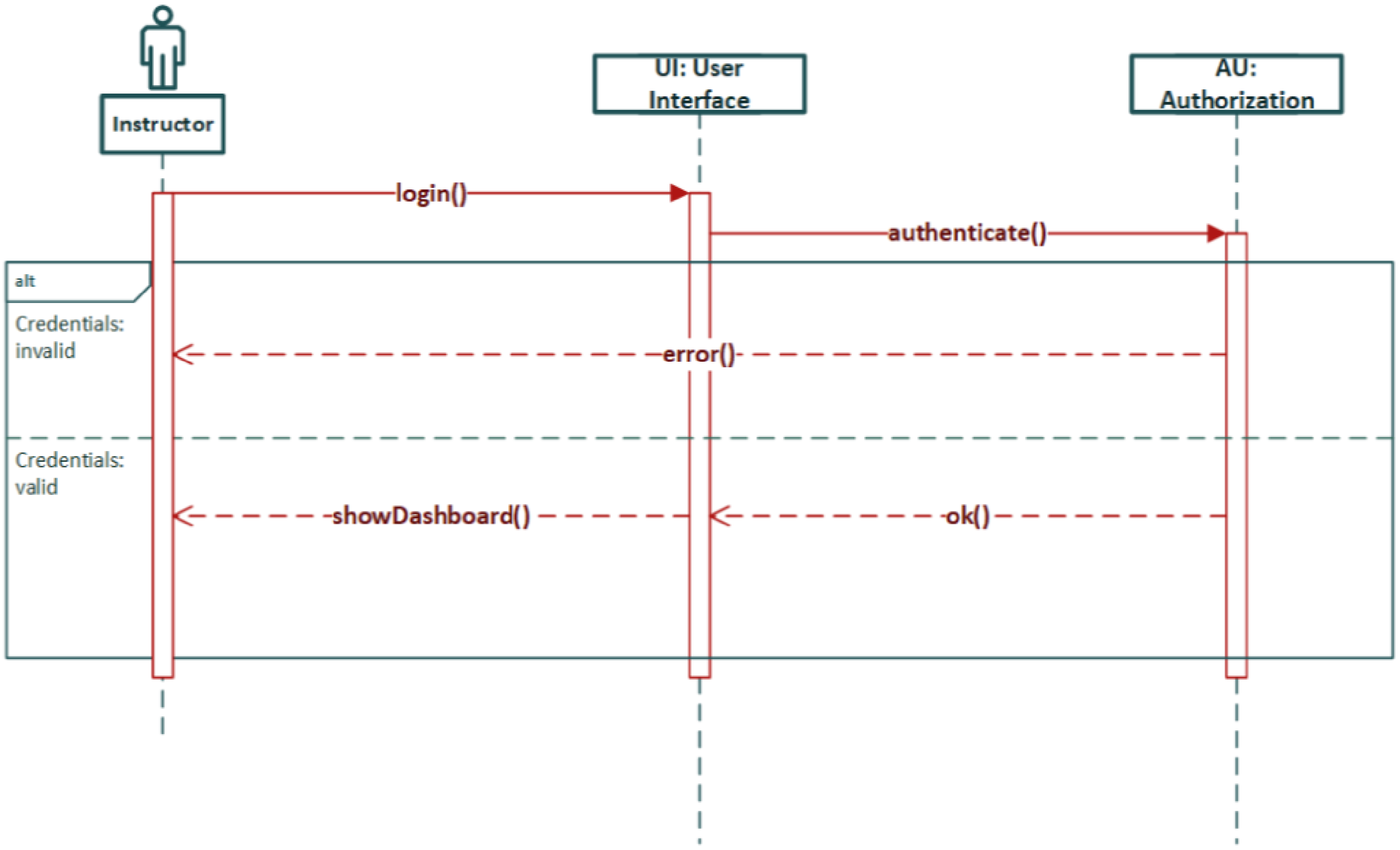


Figure 7: Sequence Diagram for the attendance session use case

Implementation

Overview

The frontend/UI of the eAttendance application was implemented using the Android framework. The backend (API) was implemented in Python programming language.

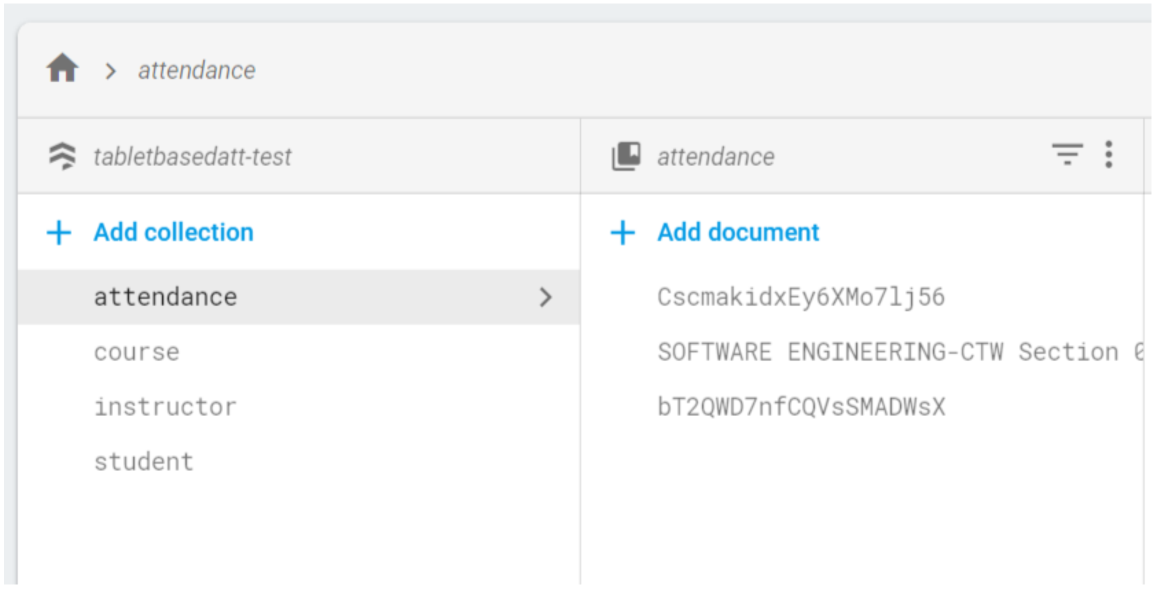
How to run

eAttendance is designed to run on Android tablets.

Steps to run the app:

1. download and install the “riskk_eAttendance_v0.1.apk” (file included in the assignment submission) on an Android phone or tablet.
2. After installation, open the app.
3. At the login screen, tap on the “Login” button (no credentials are required for this version of the app). You will be redirected to the Attendance Session screen.

Database Design Implementation



attendance
course
instructor
student

CscmakidxEy6XMo7lj56
SOFTWARE ENGINEERING-CTW Section
bT2QWD7nfCQVsSMADWsX

Figure 8: Screenshot of the database dashboard

Frontend/UI

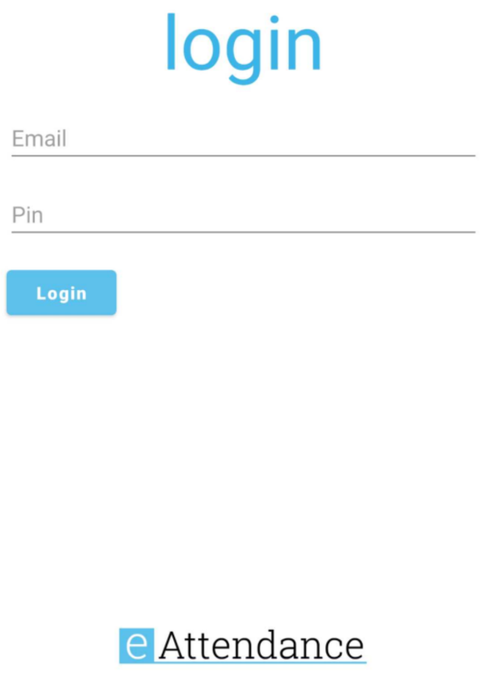
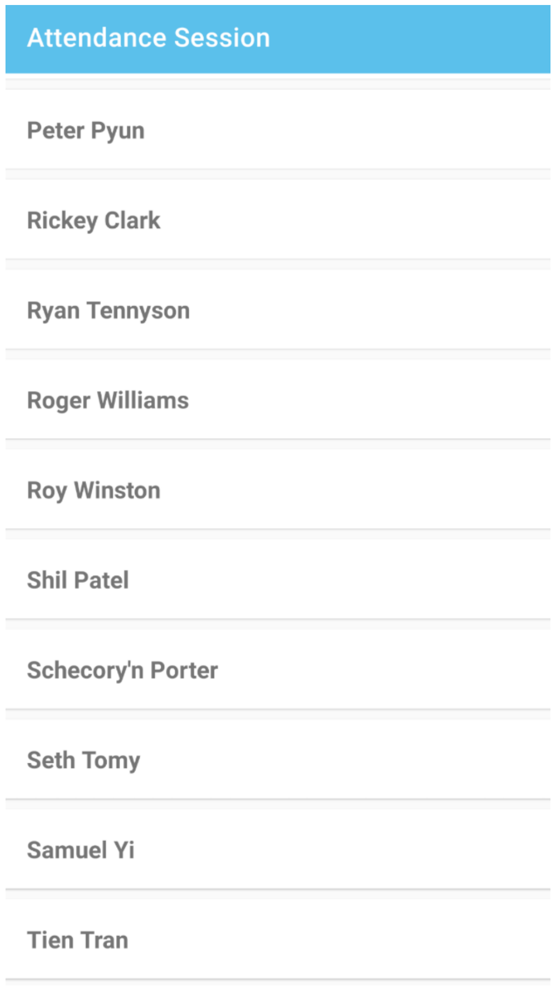


Figure 9: Screenshot of frontend implementation of the login use case



Attendance Session

Peter Pyun

Rickey Clark

Ryan Tennyson

Roger Williams

Roy Winston

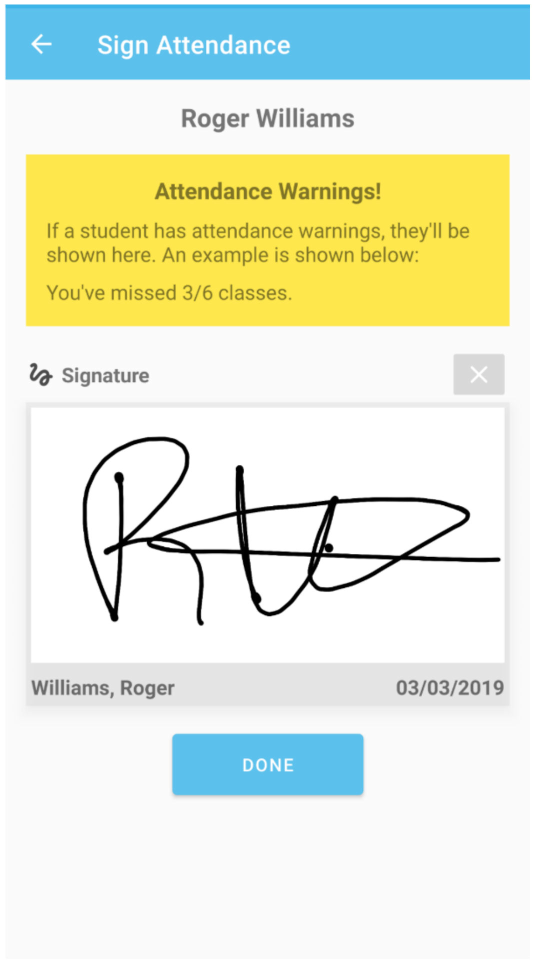
Shil Patel

Schecory'n Porter

Seth Tomy

Samuel Yi

Tien Tran



Sign Attendance

Roger Williams

Attendance Warnings!

If a student has attendance warnings, they'll be shown here. An example is shown below:

You've missed 3/6 classes.

Signature

[illegible]

Williams, Roger 03/03/2019

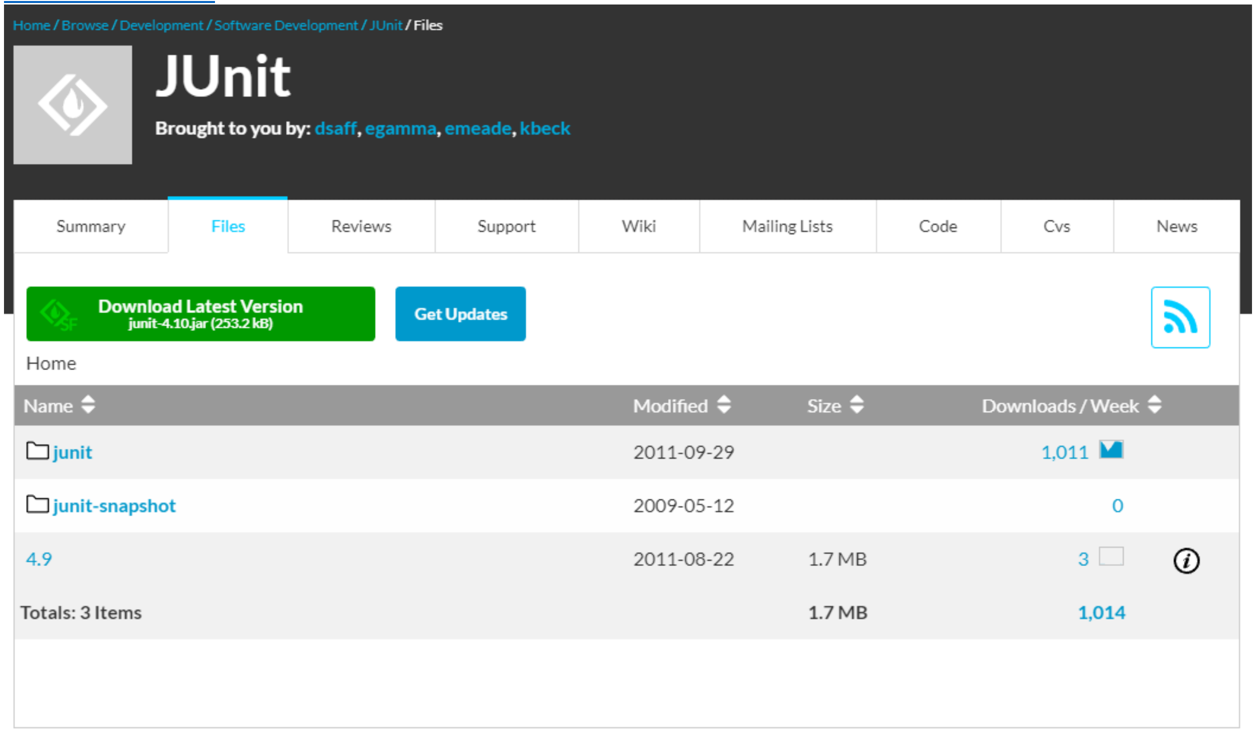
DONE

Figure 10: Screenshot of frontend implementation of the attendance session use case

Testing

How to install JUnit:

1. Download JUnit.



Home / Browse / Development / Software Development / JUnit / Files

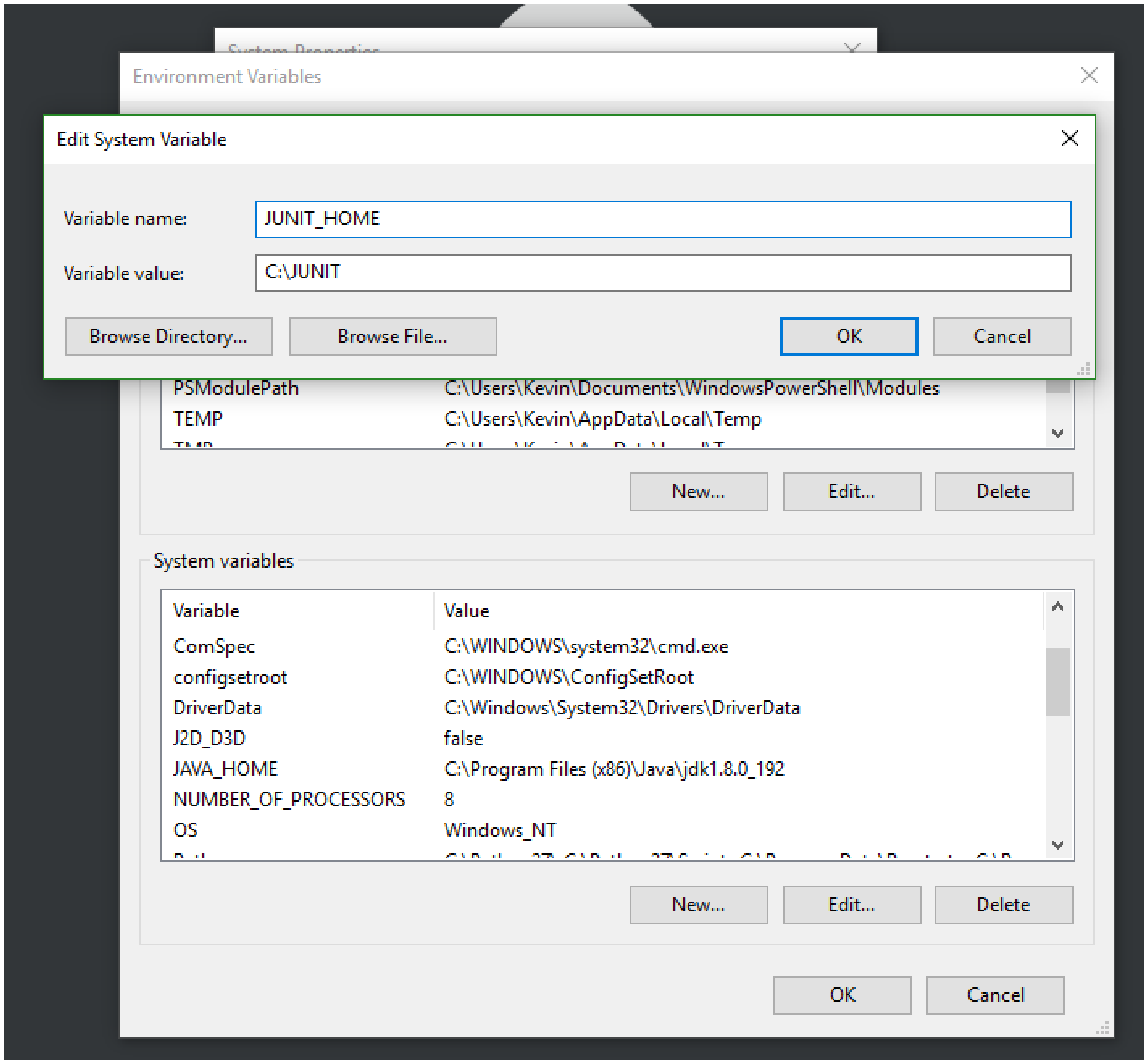
JUnit

Brought to you by: dsaff, egamma, emeade, kbeck

Home

Name	Modified	Size	Downloads / Week
junit	2011-09-29		1,011
junit-snapshot	2009-05-12		0
4.9	2011-08-22	1.7 MB	3
Totals: 3 Items		1.7 MB	1,014

2. Set the “JUNIT_HOME” environment variable to point to the *base* directory location of where your JUNIT jar file is stored. (ex: *C:\JUNIT*)



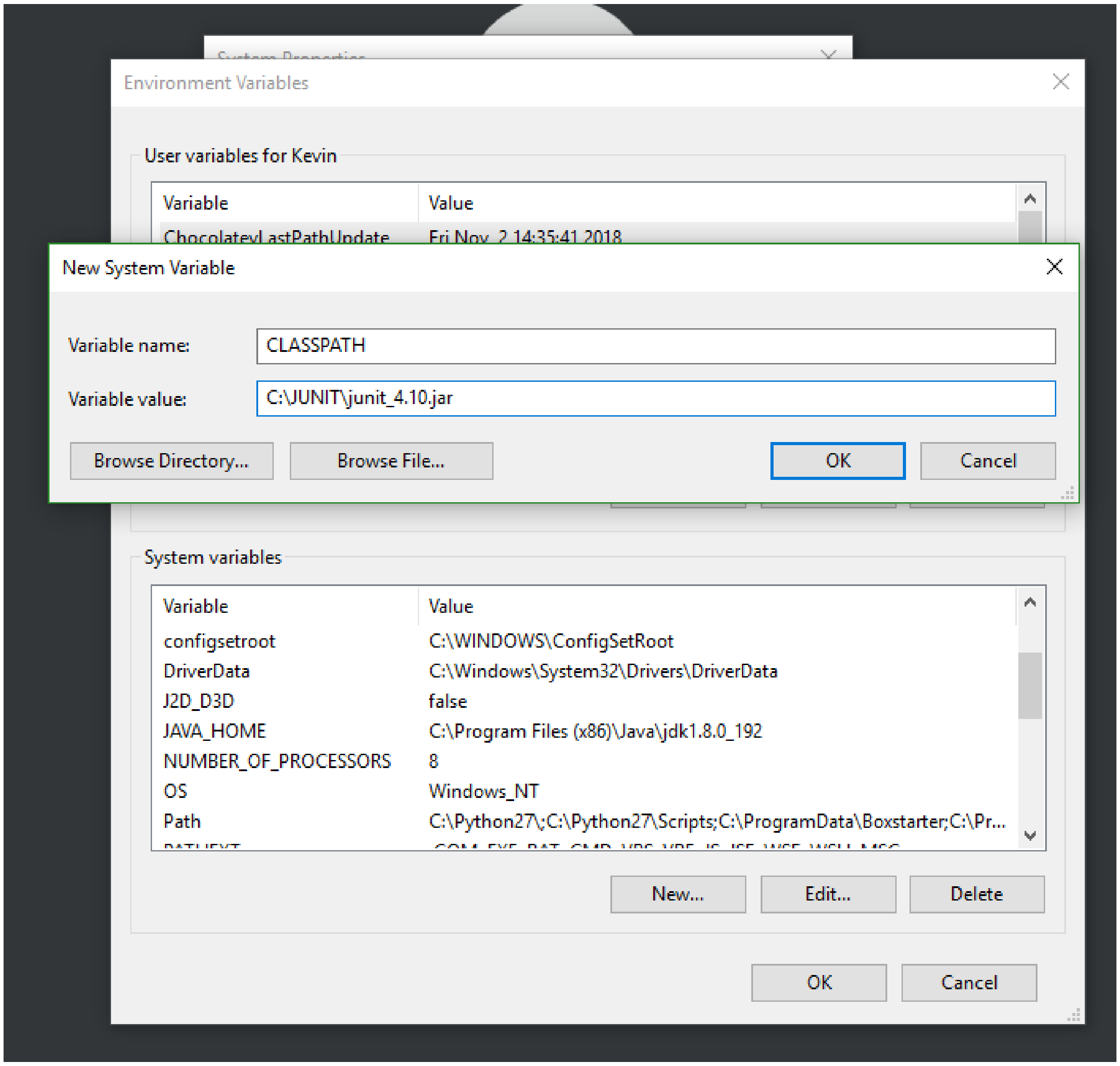
Variable name: JUNIT_HOME

Variable value: C:\JUNIT

PSModulePath	C:\Users\Kevin\Documents\WindowsPowerShell\Modules
TEMP	C:\Users\Kevin\AppData\Local\Temp

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
configsetroot	C:\WINDOWS\ConfigSetRoot
DriverData	C:\Windows\System32\Drivers\DriverData
J2D_D3D	false
JAVA_HOME	C:\Program Files (x86)\Java\jdk1.8.0_192
NUMBER_OF_PROCESSORS	8
OS	Windows_NT

3. Set the "CLASSPATH" environment variable to point to the directory location of where your JUNIT jar file is stored (*%JUNIT_HOME%\junit_**(version)**.jar*).



Environment Variables

User variables for Kevin

Variable	Value
ChocolateyLastPathUpdate	Fri Nov 2 14:35:41 2018

New System Variable

Variable name: CLASSPATH

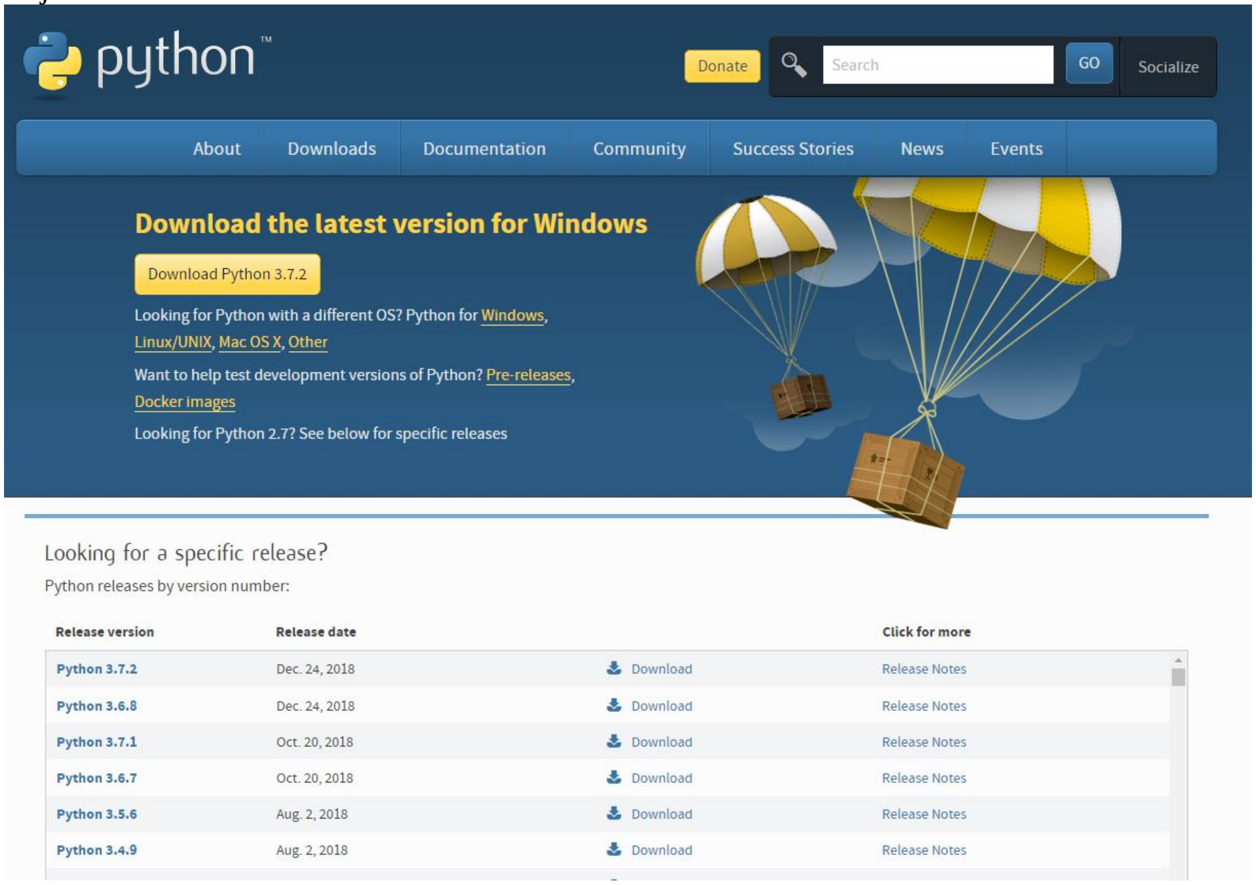
Variable value: C:\JUNIT\junit_4.10.jar

System variables

Variable	Value
configsetroot	C:\WINDOWS\ConfigSetRoot
DriverData	C:\Windows\System32\Drivers\DriverData
J2D_D3D	false
JAVA_HOME	C:\Program Files (x86)\Java\jdk1.8.0_192
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Python27\;C:\Python27\Scripts;C:\ProgramData\Boxstarter;C:\Pr...

How to Install PyUnit:

1. Install the latest version of Python. The PyUnit module comes with Python 2.1 and later so just follow the installation instructions.



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.2	Dec. 24, 2018	Download	Release Notes
Python 3.6.8	Dec. 24, 2018	Download	Release Notes
Python 3.7.1	Oct. 20, 2018	Download	Release Notes
Python 3.6.7	Oct. 20, 2018	Download	Release Notes
Python 3.5.6	Aug. 2, 2018	Download	Release Notes
Python 3.4.9	Aug. 2, 2018	Download	Release Notes

Test Case Table

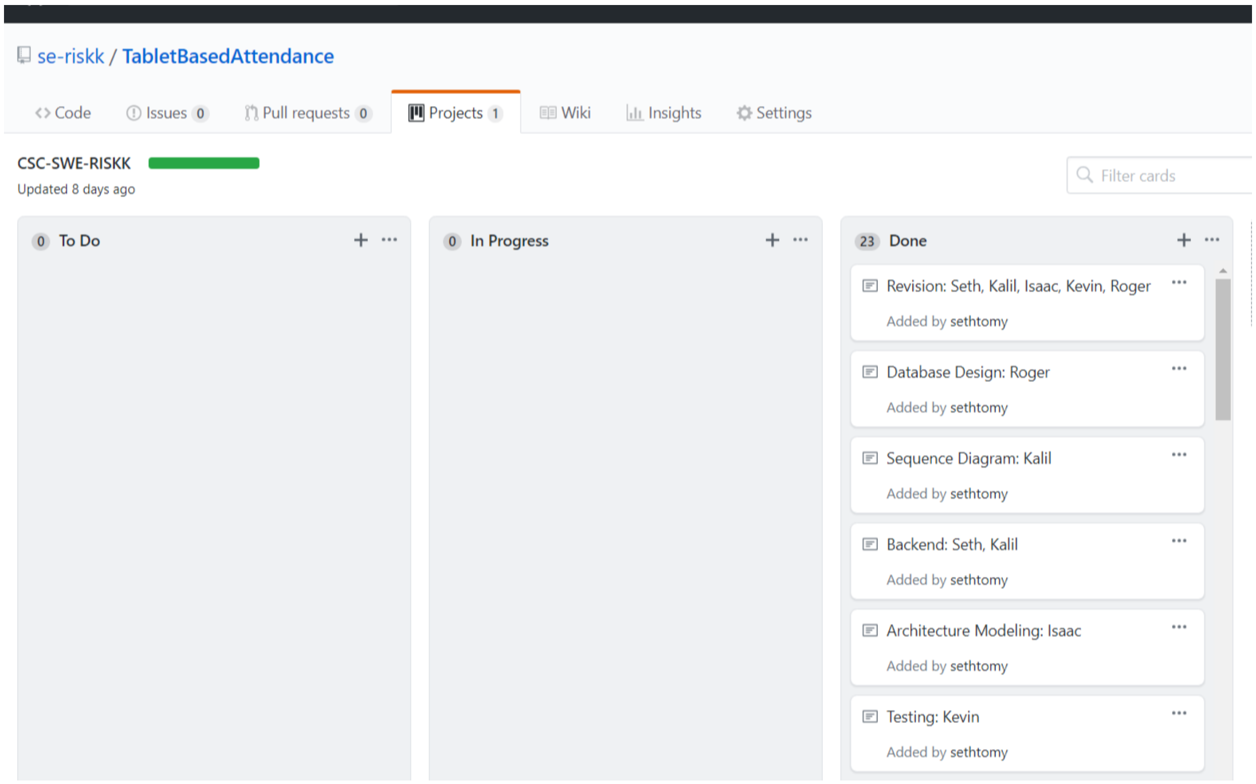
Test Scenario	Test Case	Preconditions	Procedure	Test Data	Expected Results
Instructor Login	User has entered a valid email/pin combination.	App has started and is showing the login screen	Click "Login".	Email: "amussa@gsu.edu" PIN: 1234	User is logged in using the entered email credentials
	User has entered an invalid email/pin combination.	App has started and is showing the login screen	Click "Login".	Email: "testInvalid@wrong.com" PIN: 1415	User gets an error message saying "Invalid Login"
	User has NOT entered an email.	App has started and is showing the login screen	Click "Login".	Email: (none) PIN: 1234	User gets an error message saying "Email not entered"
	User has NOT entered a pin.	App has started and is showing the login screen	Click "Login".	Email: "amussa@gsu.edu" PIN: (none)	User gets an error message saying "PIN not entered"
	User is NOT connected to the internet.	App has started and is showing the login screen	Click "Login".	Email: "amussa@gsu.edu" PIN: 1234	User gets an error message saying "Could not connect to server"
Get Class Roster's list of Students	User starts taking attendance	User is logged in and has started taking attendance for a selected class	The app will automatically get a student roster list for the selected class from the API.	Selected Class: "SOFTWARE ENGINEERING-CTW Section 003 Spring Semest"	User is shown a list of students from the selected class

Student Sign-in	User clicks a generated student name in the list	Instructor has started taking attendance and a list of students have been generated	Click a generated student name in the list.	<u>Student to be Selected:</u> “Roger Williams”	User is redirected to a student sign-in page with the student’s ID shown briefly
	User gives a signature	The user is in a student’s sign-in page	Tap and drag in signature field based on how you would normally sign in.	<u>Selected Student:</u> “Roger Williams”	Signature field will show black marks based on where (within the signature field) the user taps and drags
	User clears inputted signature	The user is in a student’s sign-in page and has given a signature	Click “Clear Signature”.	<u>Selected Student:</u> “Roger Williams” <u>Signature Field:</u> Roger Williams	Signature field is set as a completely empty signature field
	User submits signature	The user is in a student’s sign-in page and has given a signature	Click “Done”.	<u>Selected Student:</u> Roger Williams <u>Signature Field:</u> Roger Williams	Signature is stored and student attendance for this session is set to “true”
	User submits without a signature	The user is in a student’s sign-in page	Click “Done”.	<u>Selected Student:</u> “Roger Williams” <u>Student Signature:</u> (none)	User gets an error message saying “Missing Signature”

Appendix

Link to YouTube channel: https://www.youtube.com/channel/UC92oLbG_vi4123kV7LRee1A

GitHub Project page



se-riskk / TabletBasedAttendance

Code | Issues 0 | Pull requests 0 | Projects 1 | Wiki | Insights | Settings

CSC-SWE-RISKK
Updated 8 days ago

Filter cards

0 To Do

0 In Progress

23 Done

- Revision: Seth, Kalil, Isaac, Kevin, Roger — Added by sethtomy
- Database Design: Roger — Added by sethtomy
- Sequence Diagram: Kalil — Added by sethtomy
- Backend: Seth, Kalil — Added by sethtomy
- Architecture Modeling: Isaac — Added by sethtomy
- Testing: Kevin — Added by sethtomy