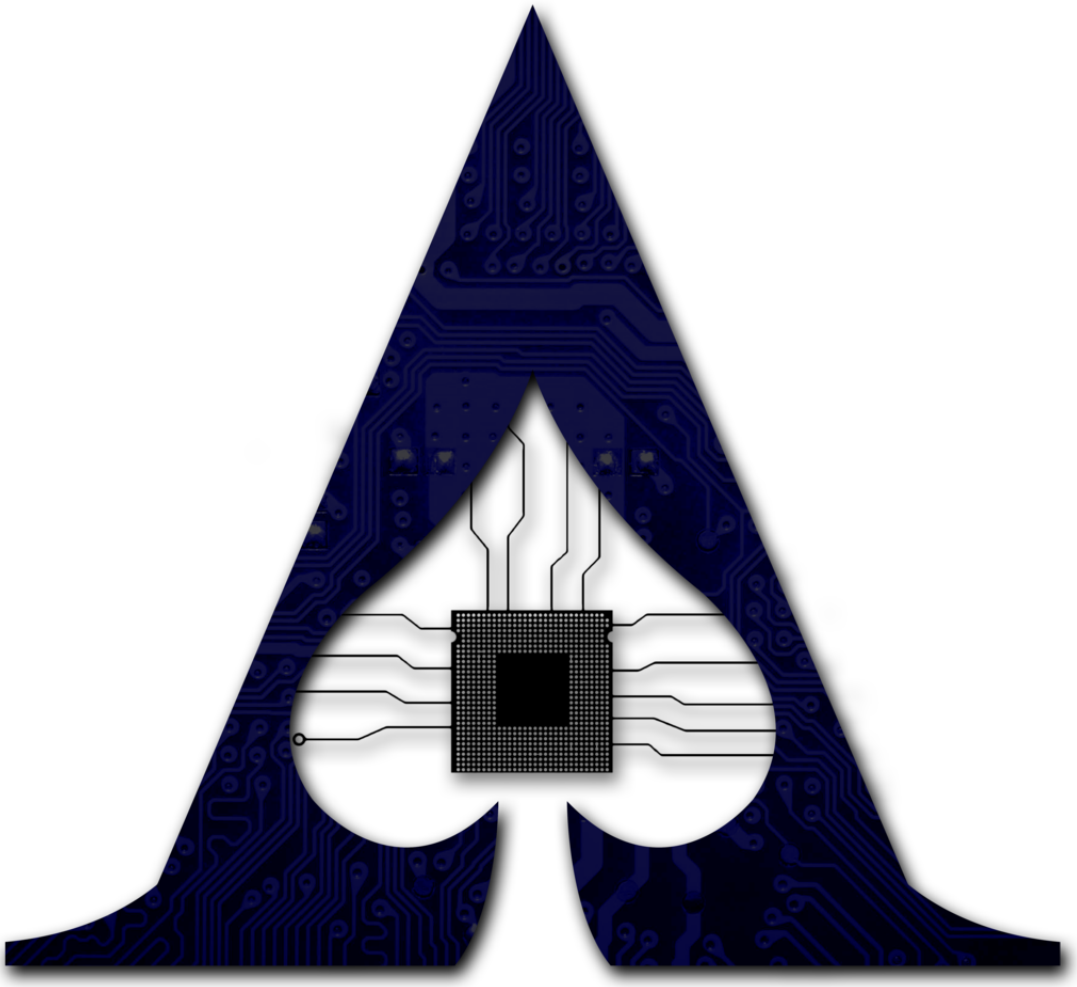
CSCC01: DELIVERABLE 3

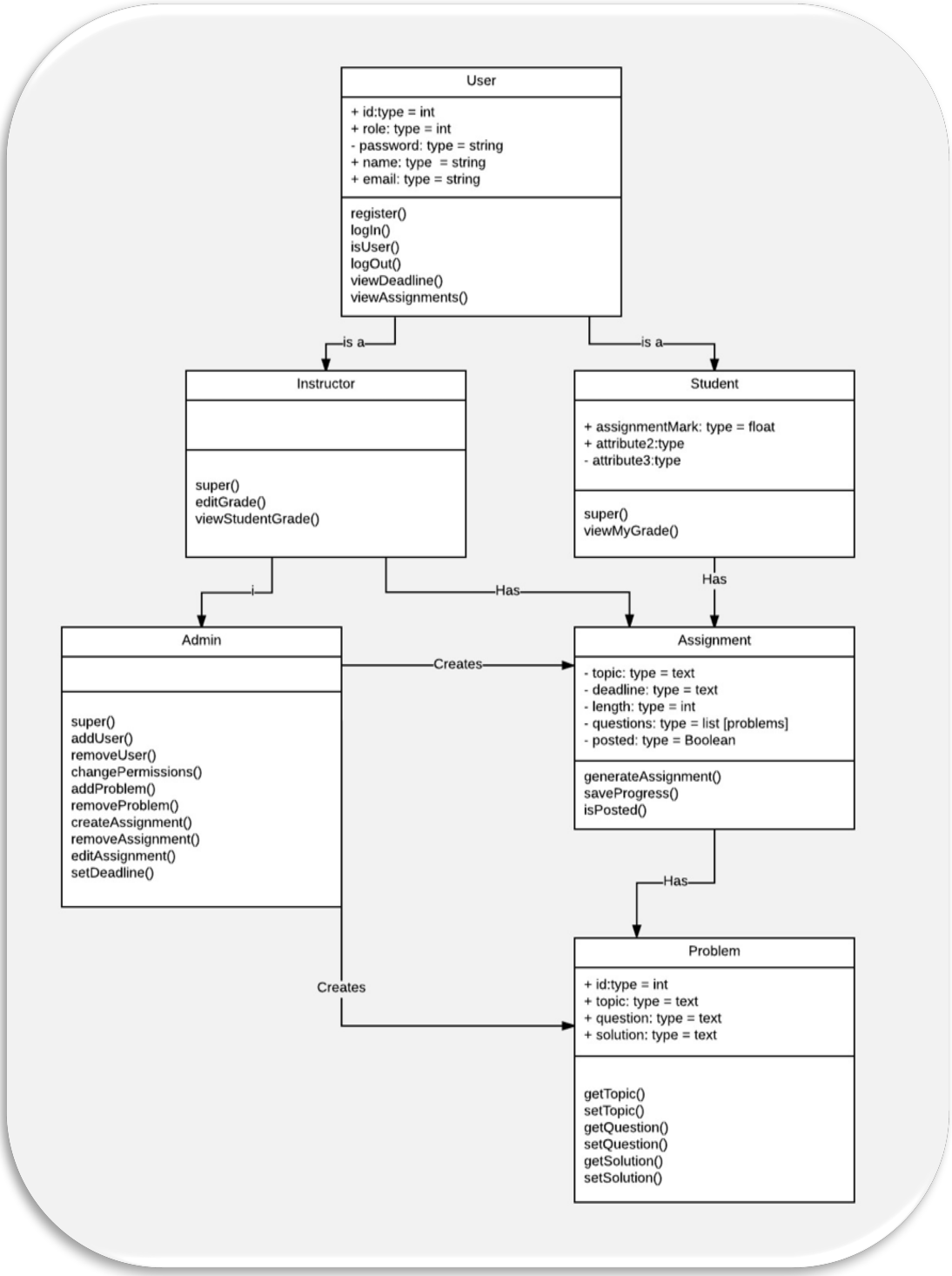
Ace of Spades



Diana Soto, Gerrit Steinbach, Bar Slutsky,
Kezaram Tharmasegaram, Jonathan Jung-Kyun An

CSCC01: DELIVERABLE 3

UML Diagram- Sprint 1 Focused on Admin



Personas & User Stories

Admin

Professor Nate River

- *Male, Mid 40s.*
- University of Toronto Professor - Architectural Design.
- Very busy with meetings, doesn't have much time in between lectures.
- Most of the grading is done by the TAs after he provides a rubric.
- Large office, but very messy with papers.
- Does not spend a lot of time with students.
- Does not hold many office hours.
- Knows how to use computers well.
- Likes smart-phones because he does not have to use the computer.
- Does not want to spend a lot of time on the computer.
- Likes marking on paper.
- Gives a lot of informative feedback on student's submissions.
- Serious about his work.
- Very meticulous about the course running smoothly, and passes off announcing/dealing with technical delays to the TA's.
- Does not want to deal with any program errors, but would have the developers answer students directly about any issues.

Professor Paco

- *Male, Mid 20s.*
- University of Toronto Professor – Computer Science.
- Speaking English and Spanish.
- Likes to make light hearted jokes, but still serious about the content of his lectures and sticking to a planned outline.
- Holds office hours for students very often, and will accommodate meeting times outside of his office.
- Doesn't have a very busy schedule, can learn the software and provide feedback.
- Advanced computer skills, doesn't need much time to get used to different software but does not want to use new systems if they create problems or slow down the course's pace.
- No preference for assignments being submitted online or on paper.

Professor Sohee

- Female, mid 50s
- Professor of Statistics at University of Toronto
- Speaks English fluently
- Not knowledgeable of advanced computer systems and operations
- Will use a software if it's simplified, and there is a way for her to get support
- Holds office hours for students as scheduled on a weekly basis
- Enjoys students engaging with their homework and would like them to have an easier tool to complete homework online

- Administers a number of courses at the University, therefore very busy schedule, relies on organized and planned routines to get through tasks per day
- Prefers to students to get immediate feedback on their homework, making marking more effiencient

As an Admin, I would like to be able to..

(Only 3 Basics Taken from Deliv2)

- As an Admin (P,N,S) I would like to log in
- As an Admin (P, N, S) add, remove, and edit problems in an existing collection for a problem set.
- As an Admin (P,N,S) add users to the software.

Language Choice

We will code the program using python. Python is a simple to use language that has many features including native GUI frameworks, and documentation is readily available. Moreover we have all used python before, and this is a language that we should all be able to use effectively.

Implementation

The implementation this far will consist of creating the database, some basic functionality to interact with the database, and the GUI to accompany this.

For the GUI we will be using Tkinter, since it comes with python, it does not require any additional downloads. Furthermore there is documentation from the official python website that we can use if we need support.

http://www.tkdocs.com/tutorial/ https://docs.python.org/2/library/tkinter.html

By the end of the sprint there should be a basic GUI that allows the user to manipulate the program, however advanced thematic features will be added later.

The Database will be written in sqlite. Sqlite is easy to implement, and it does not require a lot of learning in order for the programmer to create functions that use the database. Databases can also be created easily, and manipulated with little code. By the end of the sprint, the database should be ready to use with all functionality implemented.

https://sqlite.org/docs.html https://docs.python.org/2/library/sqlite3.html

Sprint Backlog- 26 Story Points Total

To Do:

-> Create a Database
-> Create Tables for Users and Problems
-> Create UI for a log-in screen
-> Validate account against database
-> Create individual UI for Admin, Instructor, Student respectively
-> For Admin specifically, create UI to add, remove, update - users and problems

Breakdown

1) **Add users to the software (Total = 7 points)**
   - Task 1: Create the Database for application - Bar (1 SP)
     - Create database using sqlite
       - requires learning of sqlite
   - Task 2: Create a table in database to store users – Bar (0.5 SP)
     - Dependency T1
   - Task 3: Set up back-end functions to add, remove and update problems in database. – Bar (0.5 SP)
     - Dependency T2
   - Task 4: Implement API function for querying users - Bar (2
     - Dependency T1
   - Task 5: Implement UI Add - Diana & Gerrit (3 SP)
     - Dependency None
     - requires designing of User Interface
     - requires learning Tkinter and methods
   - Task 6: Connect UI and API functionality - Bar, Gerrit, Diana (2 SP)
     - Dependency T3& T4
     - UI programmers have to work with the backend to connect the tasks

2) **Add, remove, and edit problems from question bank (Total = 12 SP)**
   - Task 1: Create a table in database to store problems - Kez, Jon (1 SP)
     - Dependency None
     - requires learning sqlite
   - Task 2: Implement API function for querying problems - Kez, Jon (2 SP)
     - Dependency T1
   - Task 3: Implement UI - Diana & Gerrit (8 SP)
     - Dependency None
   - Task 4: Connect UI and API functionality - Kez, Jon, Gerrit, Diana (2 SP)
     - Dependency T2 & T3
     - UI programmers have to work with the backend to connect the tasks

3) **Log in (Total = 7 SP)**
   - Task 1: Implement API functions to Validate user against database upon log-in (2 SP)
     - Dependencies: Story 1 Task 1,2
   - Task 2: Implement UI LogIn and Home-Screen (3 SP)
     - Dependencies None
   - Task 3: Connect UI and API functionality (2 SP)
     - Dependency T1 and T3
     - UI programmers have to work with the backend to connect the tasks

Burndown Chart