<CapstoneReservation> Milestone 1 Summary

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Steve	SteveLocke	21
Andres	anunezze	21
Edward	xTEddie	21
Fred	fredericDaigle	21
Kammy	kammywong609	21
Ken	kemoua	21
Garrison	garrisonblair	21

Project summary

Capstone reservation is a project to improve the existing capstone room booking system to support self-serve room reservation for Concordia University ENCS students and capstone room management for Capstone Technician. The main features are booking capstone room online or using card reader, room booking rules management and room usage management and reporting.

Risk

- Integrating the system with AITS databases. AITS is the Concordia ENCS departments IT support who maintains records of all ENCS students. These records will be used to verify that users of the system are indeed students of the ENCS department. This is a high risk because it involves an external system that is not usually accessed by outside sources and that is not in our control. This risk is handled in the second iteration in this user story.

- The customer wants every booking to be backed up to the previous system when this system starts being used in case of any issues that might lead to needing to use the old system again. Doing this will require interfacing with the WEBCalendar application in a way it was not designed for. For this reason it will be quite risky to do this and is handled in iteration 2 in this user story

Legal and Ethical issues

- Only ENCS students is eligible for using capstone rooms
- Student login info must be secured
- Although we have access to hashed ENCS account passwords through the AITS database, we are not permitted to use them.

Velocity

Project Total: X stories, X points over X weeks

Iteration 1 (0 stories, 0 points)
Gathered Requirements and investigated whether WEBCalendar would be reused or not.

Iteration 2 (2 stories, 21 points)
Student login and Room booking are created to ensure the minimum requirement for a running system.

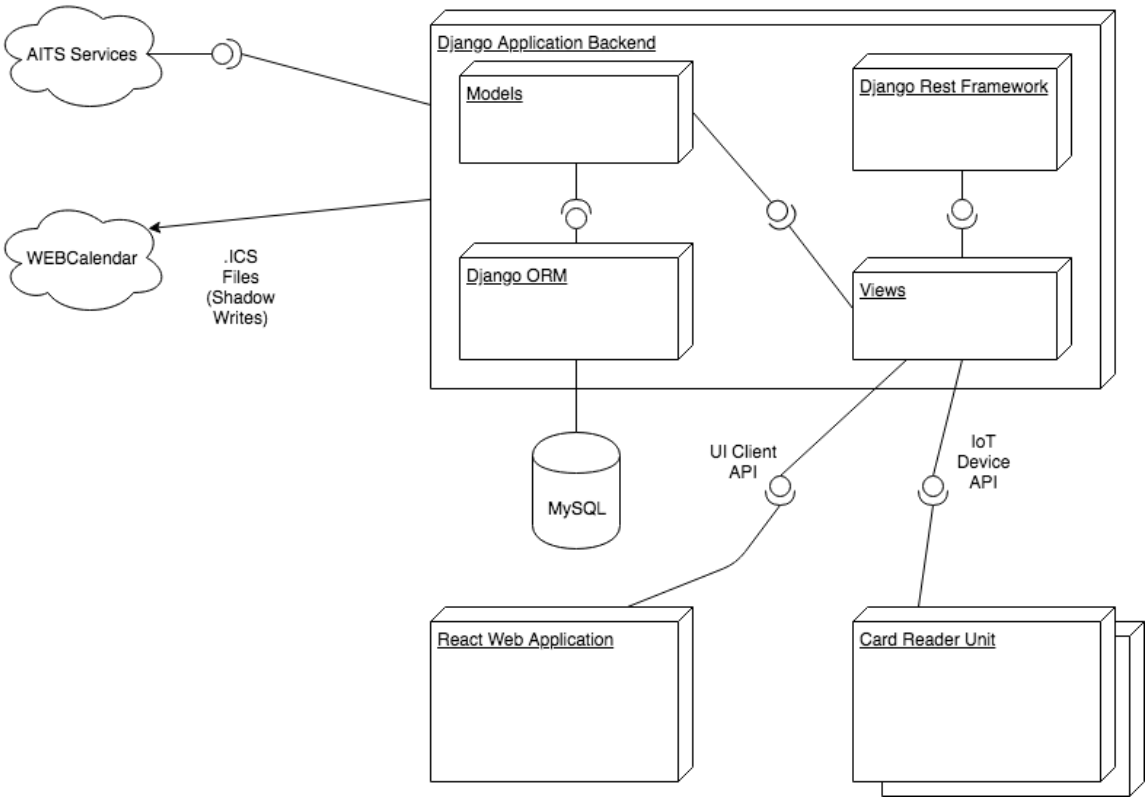
Iteration 3 (4 stories, 44 points)
Max four sentence paragraph describing main achievements.

Release 1 Total: 11 stories, 98 points over X weeks
Release 1 aka Iteration 4, (5 stories, 33 points)
Max four sentence paragraph describing main achievements.

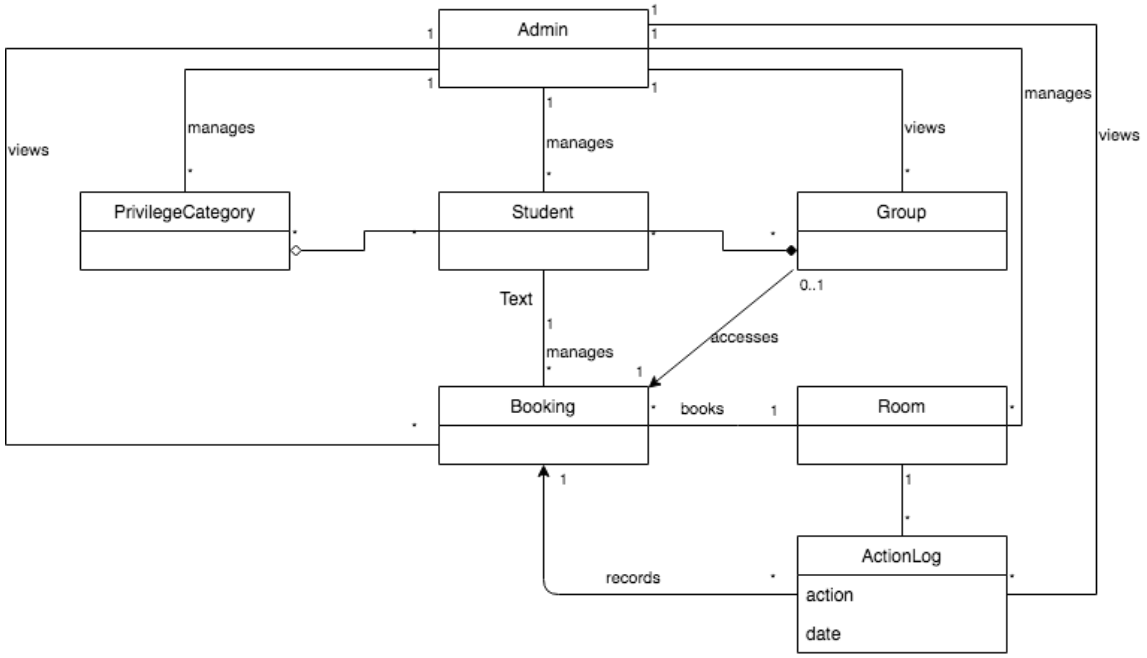
Release 1 aka Iteration 5, (3 stories, 34 points)
Max four sentence paragraph describing main achievements.

Overall Arch and Class diagram

High Level Architecture



Domain Model



Infrastructure

Django

Django is a Python web application framework. It features an ORM, request routing mechanisms and a templating engine, although the latter is not used in this Project. It will manage the database and perform object mapping to tables that it generates based on class definitions of models. A Django application will be the central component of this system. It will expose a UI Client API and an IoT Device API.
This framework was chosen because it enables rapid and flexible development of a backend system and the team has experience developing with it.

Django Rest Framework

This is an additional library that can be used with Django which facilitates the creation of REST API's. It provides many utilities for managing requests and responses, or to serialize and deserialize objects into JSON, among many other features.

React

React is a frontend javascript framework used to develop single page applications. It is the most used javascript framework nowadays. It is very well supported in the community. This will be used to implement the user inferface and will communicate with the backend server over a REST API.

SQLite

SQLite is a lightweight database engine which requires little setup and is therefore a perfect fit for a development environment. This project uses it in the development build to avoid needing to setup a full MySQL database in the local environment. It is supported by Django and is the default database engine Django apps use.

MySQL

MySQL is a popular database management system that is reliable and performant. It is the only system that is supported by Django and that is available on the AITS infrastructure where this system will run during production.

Jest

Jest is a UI unit testing framework for ReactJS projects. It is used to make snapshots and to ensure that the interfaces rendered corresponds to what is expected. It is easy and fast to use. If any changes occur in the UI, it is easy to track with the snapshots.

Name Conventions

PEP8
AirBnB Javascript Style Guide

Code

Key files: top **5** most important files (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or are a prototype so we grade these accordingly.

File path with clickable GitHub link	Purpose (1 line description)
server/apps/booking/views/booking.py	Contains the logic for the Booking entity CRUD API.
server/apps/rooms/views/room.py	Contains the logic for the Room entity CRUD API
server/apps/booking/models/Booking.py	Booking model definition.
client/src/components/Calendar/index.js	Contains the UI for the Calendar
client/src/components/ReservationDetailsModal/index.js	Contains the UI for the Reservation Details Modal

Testing and Continuous Integration

List the **5** most important test with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)

server/apps/booking/tests/testBookingAPI.py	Tests the Booking API
server/apps/booking/tests/testBooking.py	Tests the Booking model
server/apps/rooms/tests/testRoomAPI.py	Tests the Room API
client/src/tests/calendar/calendar.test.js	Tests the Calendar UI (Snapshots)
client/src/tests/requestHeaders.tests.js	Test requestHeaders.js (unit tests)

Continuous Integration

Jenkins is the chosen tool for performing CI. We have chosen to build a declarative pipeline which allows us to programmatically control the behaviour of our pipeline through code rather than strictly through the graphical user interface settings of the jobs. Our declarative pipeline functions using a Jenkinsfile and a bash script called jenkins.sh which are added to the Git repository of each branch. The Jenkinsfile is written is Groovy and controls the flow of the builds and navigation between stages of the build process, which we store more complex behaviour and scripts within the jenkins.sh file which is written in Bash, which allows for better readability and configuration when necessary.

We currently have one job which runs whenever there is a push to the remote repository from a local branch which functions using polling and webhooks. Our pipeline consists of several stages including the checkout scm, static analysis, build, unit tests, and integration tests.

We have added an integration with slack to notify us of the status of each Jenkins build within a dedicated slack channel, indicating the branch and status of the build on the branch, in order to be notified of the status immediately. As we are unable to add the Jenkins check during pull requests, our integration with slack allows the reviewers of the pull requests to verify the quality of the pull request simply by checking the status of the build within the channel.

At this time, the static analysis and integration tests are just placeholders until we develop integration tests and have time to add static analysis in the next iteration. We will also look to adding code coverage. As the production environment is not set up, the application is not deployed. We will explore deployment at a later iteration.