TEAM 2 PROJECT REPORT

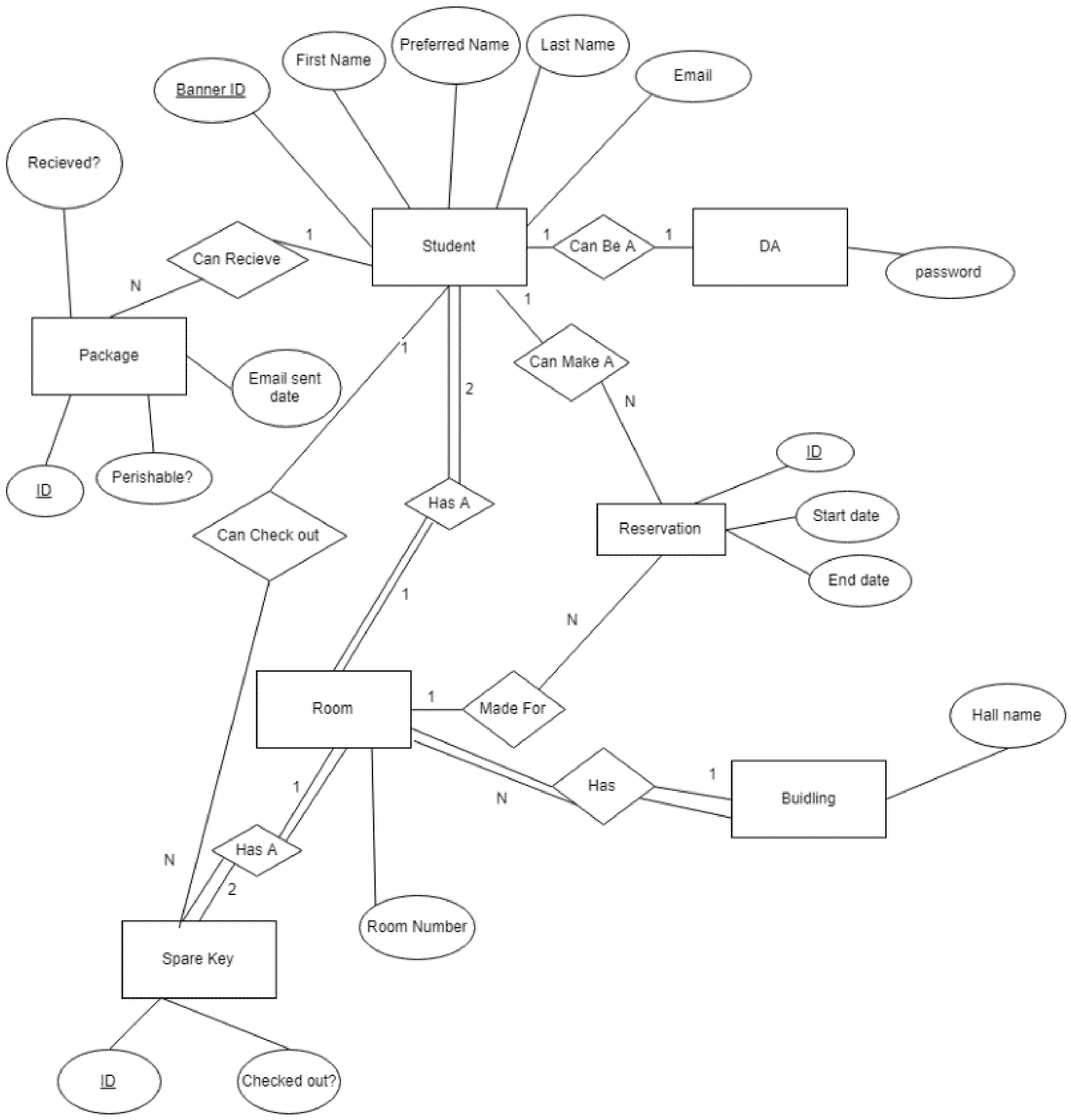
DESCRIPTION

Our database application is a housing management system, meant to work how an actual digital housing management system for ASU would. One of our team members, Jordan Fry, is a desk assistant for ASU Housing, and came up with the idea for the project out of frustration with the current system. At present, all student information is stored in two Excel spreadsheets, emails about incoming packages must be sent manually, DAs must perform guesswork to see if they have the right student, and in general, things are far more inefficient than need be.

Our project aims to remedy this. Originally, we intended this to actually be used by Housing. However, after sending an email to the head of the department, we were told that such a project was already in progress. We decided to move ahead with the idea as our class project anyway.

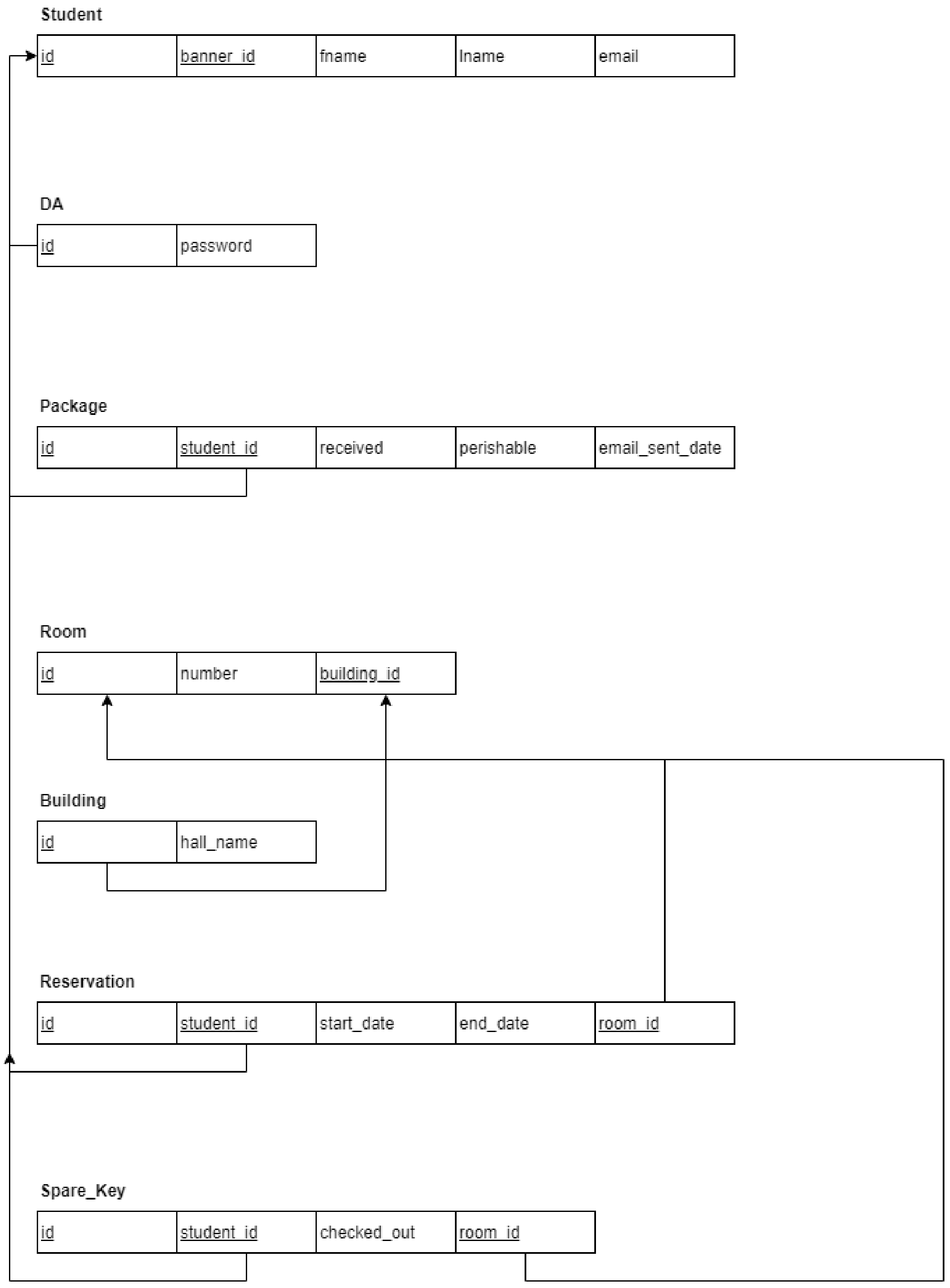
As detailed below, the application includes mechanisms for managing packages, spare keys, and room reservations. These constitute the real tasks that ASU Housing DAs perform on a day-to-day basis.

ER DIAGRAM



Note that due to how Rails sets up the database, each entity has an id, date_created, and date_modified field automatically. Thus there are no weak entities.

RM DIAGRAM


Student
id
banner_id
fname
lname
email
DA
id
password
Package
id
student_id
received
perishable
email_sent_date
Room
id
number
building_id
Building
id
hall_name
Reservation
id
student_id
start_date
end_date
room_id
Spare_Key
id
student_id
checked_out
room_id

DATABASE POPULATION

The application we have built is intended to be used by ASU Housing desk assistants, who do not have the authority to alter student or residence hall information. As such, and in lieu of access to ASU's actual student database, we have opted to pre-populate the database with residence halls, rooms, and students.

Some concessions have been made in the name of of simplicity. Each residence hall is assumed to have exactly four floors, with exactly 20 rooms each. Example valid room numbers would be 101, 219, or 410. 501 or 121 would be invalid. Additionally, many of the students have fake names and emails, and all BannerIDs are fake for obvious reasons.

SAMPLE SQL QUERIES

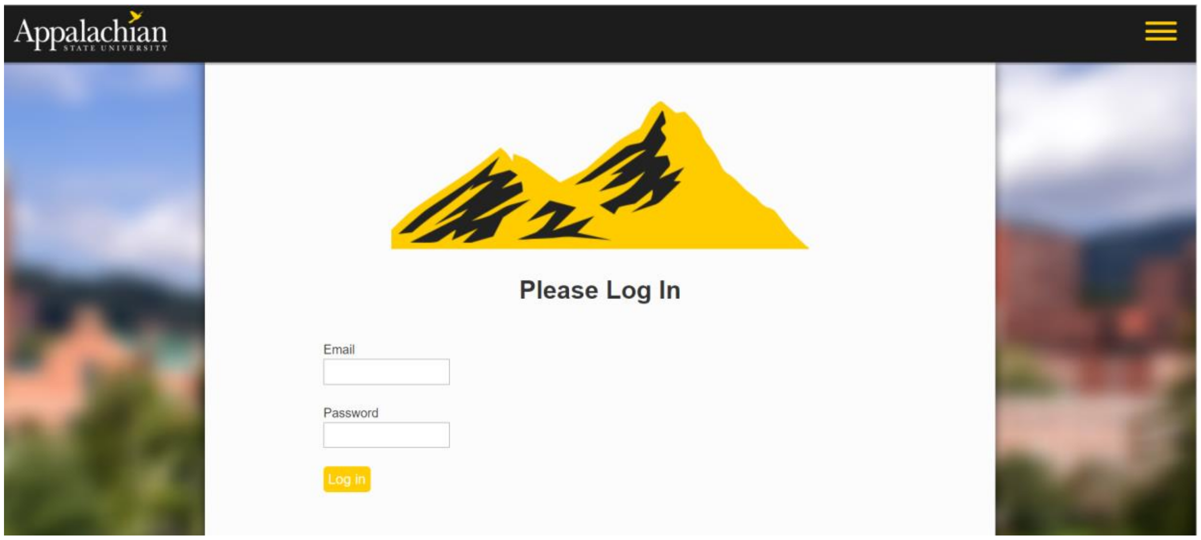
Here are a handful of SQL queries we used across the project. Note that this is only a sample.

```sql
SELECT S.lname, S.banner_id, R.start_date, R.end_date FROM students as S join
reservations as R on (S.id=R.student_id);
SELECT fname, lname, email, banner_id FROM students WHERE (lname LIKE
'%[search]%');
UPDATE packages SET received = TRUE WHERE id = [package id];
DELETE FROM reservations WHERE id = [reservation id];
```

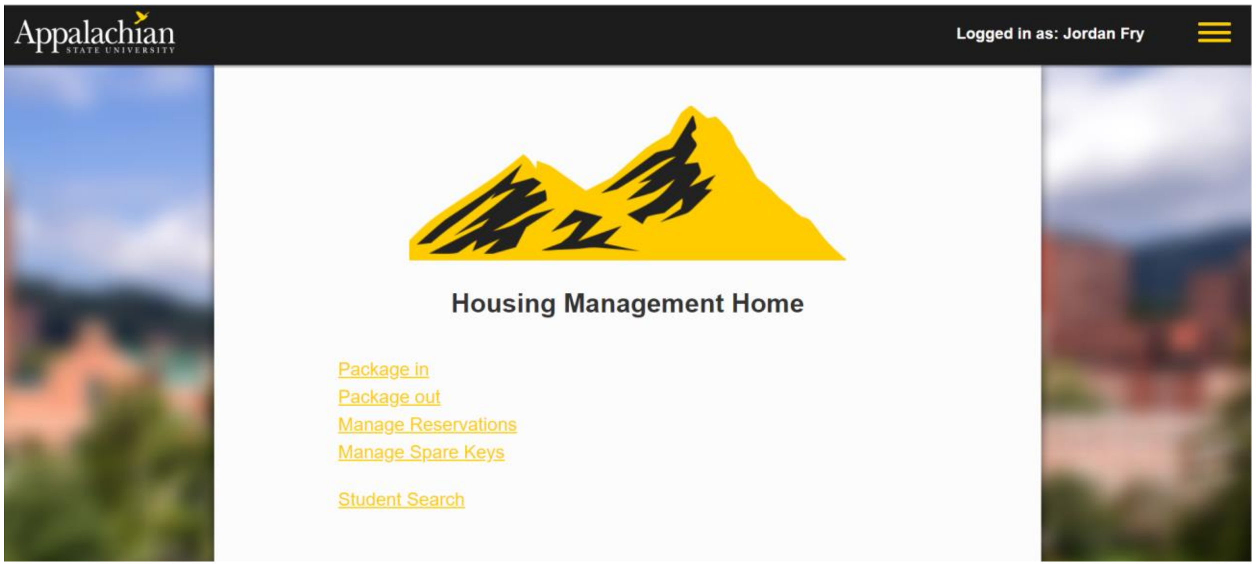
TECHNOLOGY OVERVIEW

In place of PHP, we opted to use Ruby on Rails for our project. On the client side, we used Vue.js. Other technologies used included the bcrypt Ruby gem for hashing and salting passwords, SCSS (an enhanced version of CSS) for styling, and Git for version control.

APP: LOGIN AND HOMEPAGE

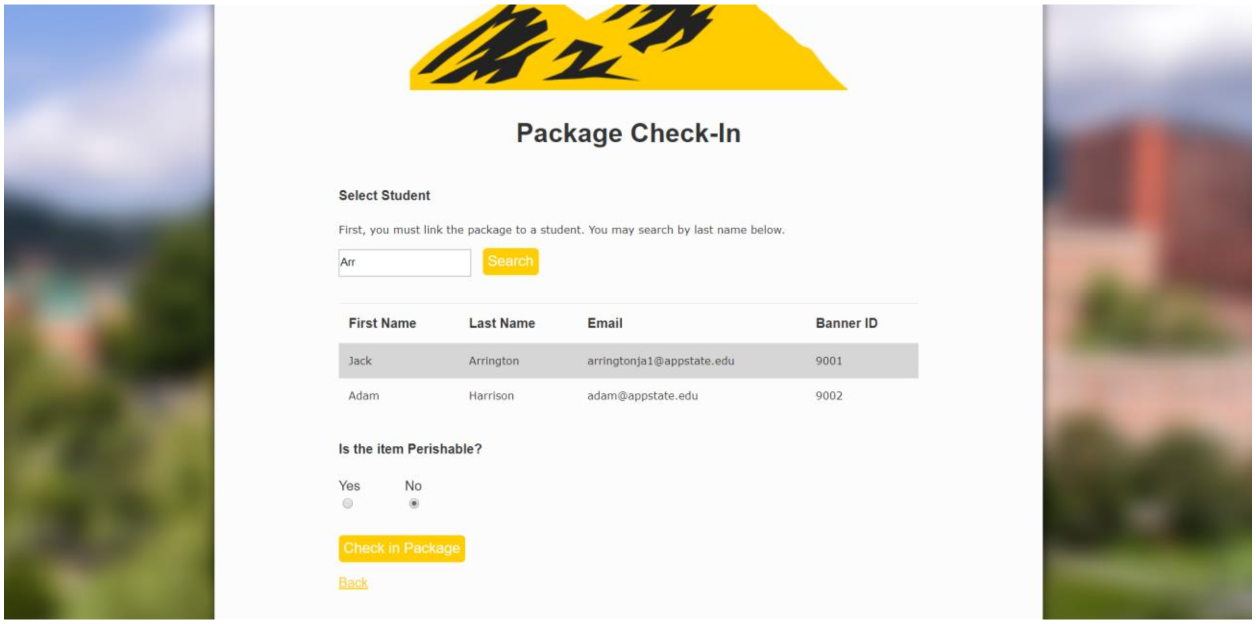


The above screen is shown to users upon launching the app. You can login with the email fryja@appstate.edu and the password **foobar**. This will create a session that will persist until you logout. Note that passwords are hashed and salted.



You will now be on the homepage. The linked pages are detailed below.

APP: PACKAGE IN



Package Check-In

Select Student

First, you must link the package to a student. You may search by last name below.

Arr

First Name	Last Name	Email	Banner ID
Jack	Arrington	arringtonja1@appstate.edu	9001
Adam	Harrison	adam@appstate.edu	9002

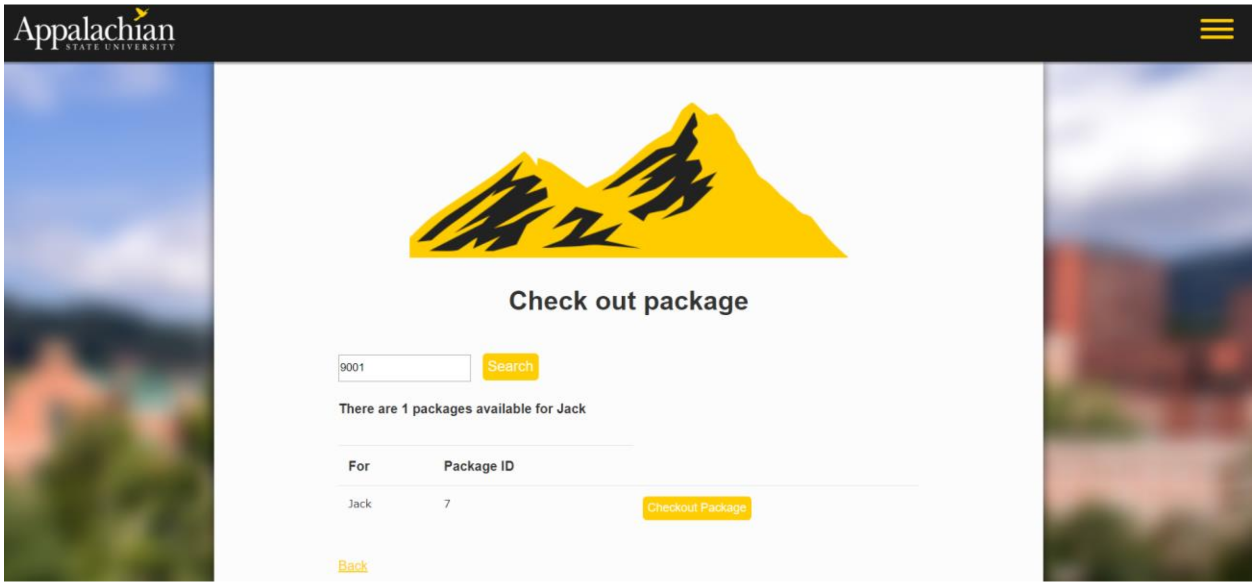
Is the item Perishable?

Yes [✓ No]

On the package check-in page, we can search for a student by their last name. This will send an AJAX request to the server and return students with similar last names. From there, we can select which student we want and whether or not the item is perishable, then press the check in button. This will send another AJAX request and automatically update the database.

An email will also be sent to the student, informing them they have a package.

APP: PACKAGE OUT



Check out package

9001

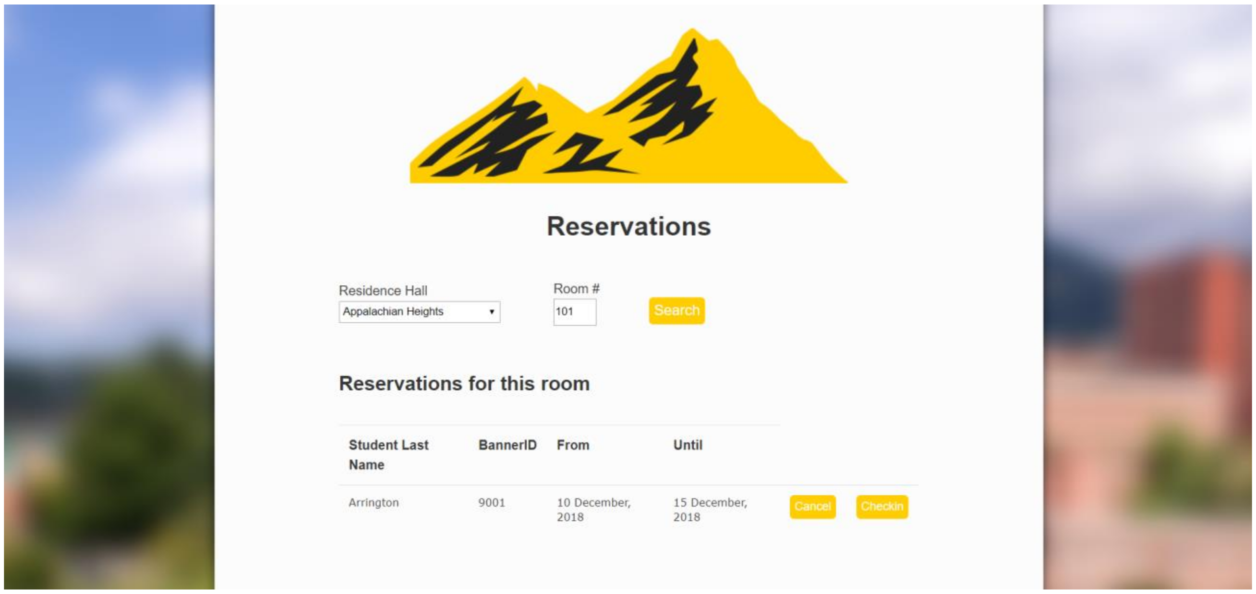
There are 1 packages available for Jack

For	Package ID	
Jack	7	Checkout Package

On the check-out page, we can enter a student's BannerID and see any packages they have waiting. The reason we chose to enter the BannerID, rather than have a search by name, is because in the real world there would be an AppCard swipe to ensure the student is who they say they are. The card swipe technology used by ASU does nothing more than paste the BannerID of the card into the currently selected form on the PC, so our application is compatible with the existing system.

Once we have entered a BannerID, we can check out any packages shown. This will send an AJAX request telling the server to mark the package as received. Once the server has responded saying the transaction was successful, the package will automatically disappear.

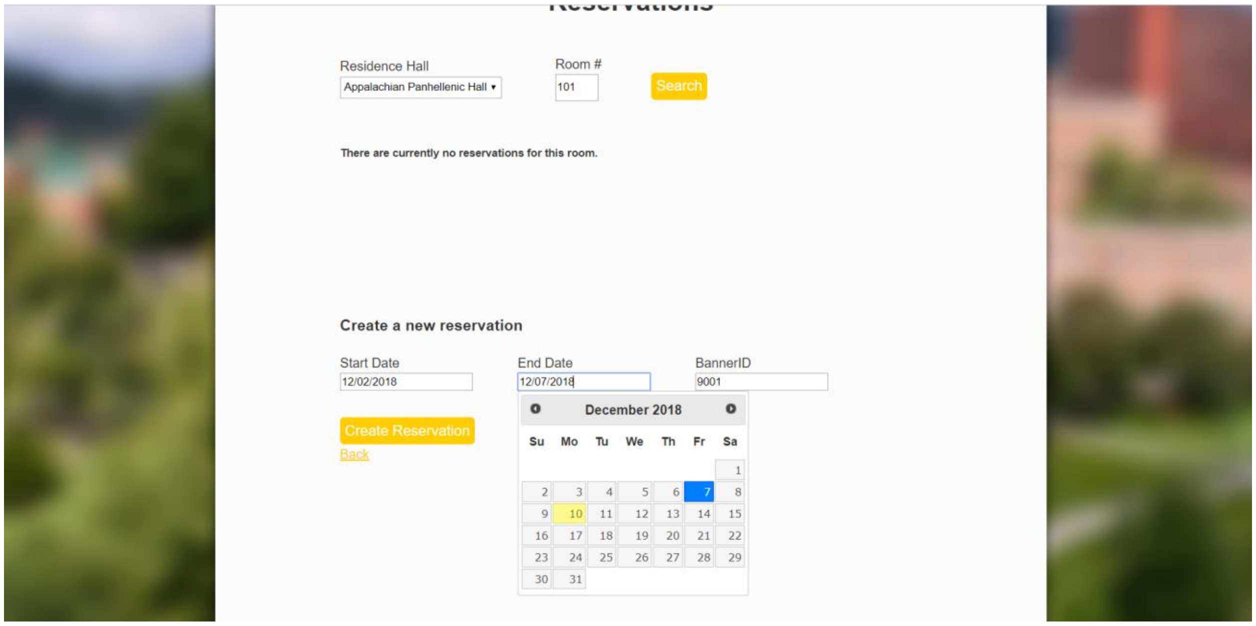
APP: MANAGING RESERVATIONS



Reservations for this room

Student Last Name	BannerID	From	Until
Arrington	9001	10 December, 2018	15 December, 2018

On the reservations page, after entering a hall name and room number, we can view any existing reservations on the room, and either cancel, check in, or (if they have already been checked in), check out those reservations. As before, there is no need for the page to reload, as these requests are handled with AJAX.



Residence Hall: Appalachian Panhellenic Hall ▾ Room #: 101 Search

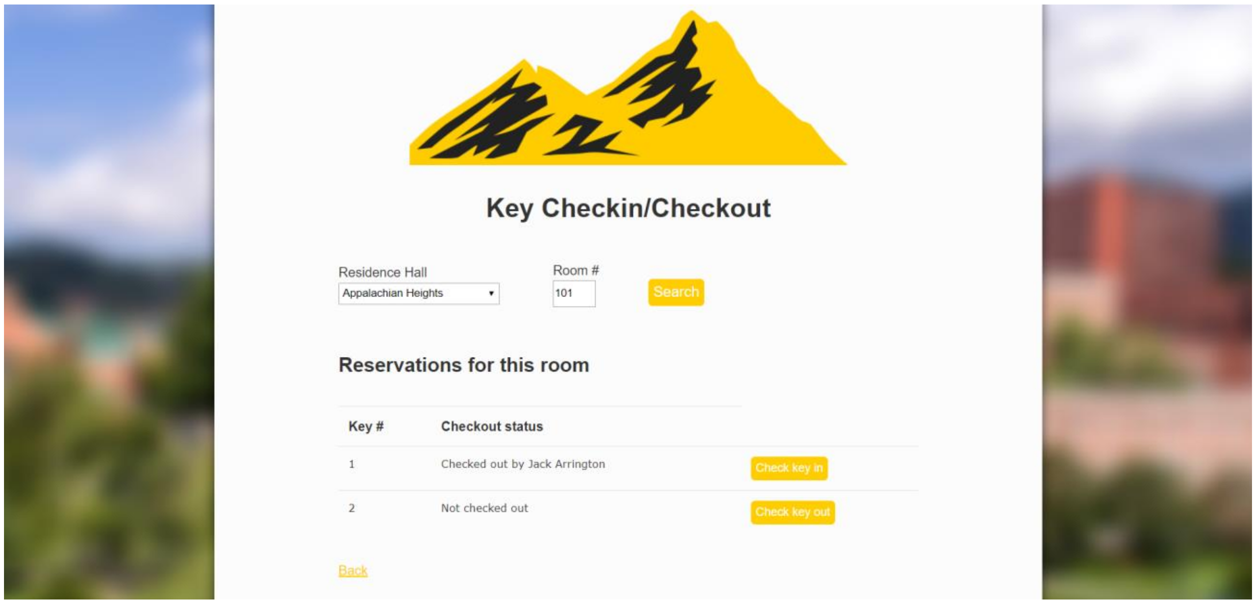
There are currently no reservations for this room.

Create a new reservation

Start Date	End Date	BannerID
12/02/2018	12/07/2018	9001

Scrolling down on the same page, we can create a new reservation by selecting start and end dates and entering a BannerID for the student wanting to make the reservation. **Constraints** are applied so that the dates cannot overlap with existing reservations, the start date cannot come after the end date, etc.

APP: MANAGING SPARE KEYS



Key Checkin/Checkout

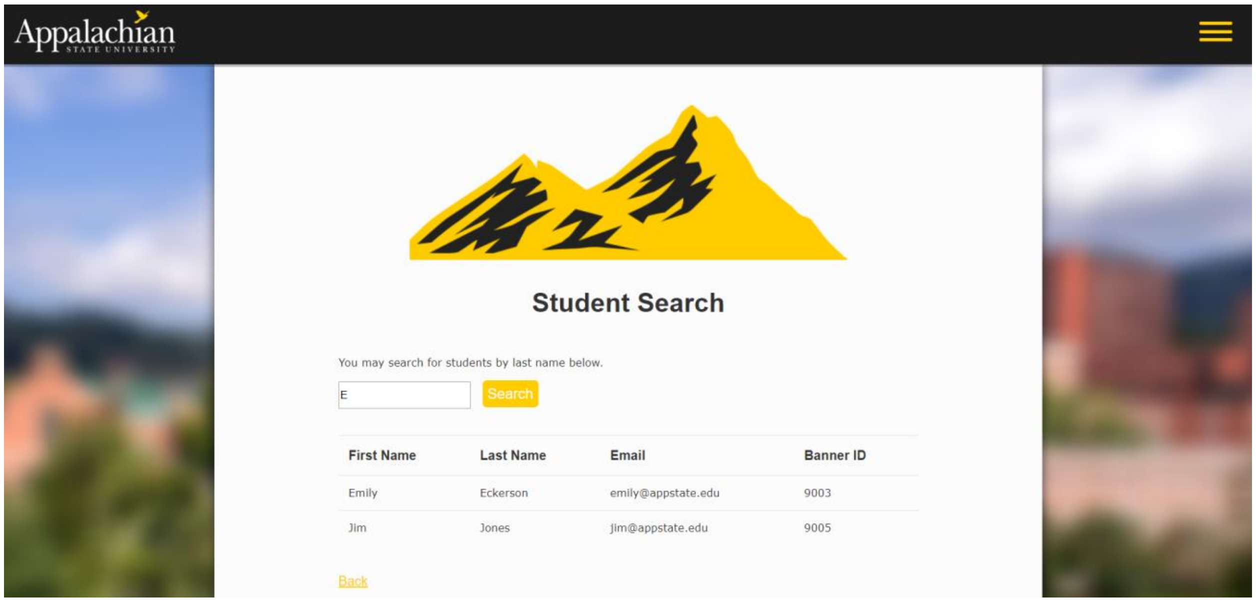
Residence Hall: Appalachian Heights ▾ Room #: 101 Search

Reservations for this room

Key #	Checkout status	
1	Checked out by Jack Arrington	Check key in
2	Not checked out	Check key out

Similar to reservations, we can manage spare keys by selecting a hall name and room number and searching. Once that room has been found, it will show the current status of the two spare keys. If one has been checked out and the student wishes to return it, it can be checked back in. If a student wishes to check out a key, the check out button can be pressed and the user will be prompted to enter a BannerID. The appropriate request is sent to the server and the form is updated automatically.

APP: SEARCHING FOR STUDENTS



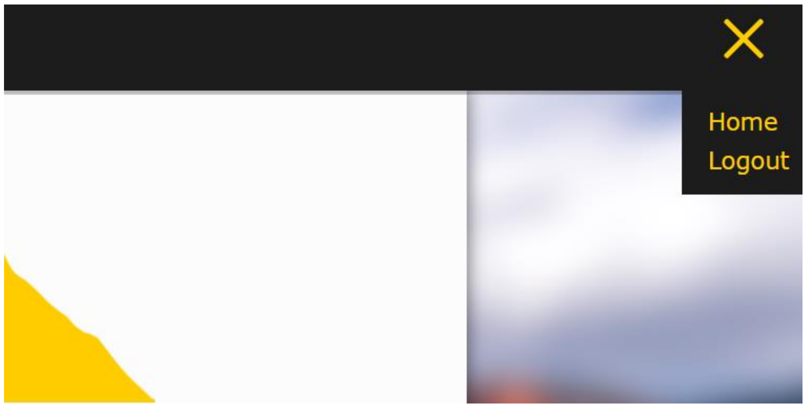
Student Search

You may search for students by last name below.

First Name	Last Name	Email	Banner ID
Emily	Eckerson	emily@appstate.edu	9003
Jim	Jones	jim@appstate.edu	9005

Since we did not have an actual card swiper handy, we thought it would be wise to include some functionality to search for a student. The student search page allows you to search for a student and displays their first and last names, email address, and BannerID. Other fields are filtered out for security purposes.

APP: LOGGING OUT



When you are done, the hamburger menu in the top right corner will display a dropdown menu with the option to log out. This will end your session.

PARTICIPANTS

Our team consisted of Jack Arrington, Jordan Fry, Robbie Rothrock, and Greg Gosnell.

LINKS

Our project can be located at http://dbproject.jackarrington.com. We opted to use a personal domain rather than student2, as student2 does not easily allow us to deploy a Ruby on Rails project.

The Github page for the project, with all code, can be found at https://github.com/StarScape/DB-Project.