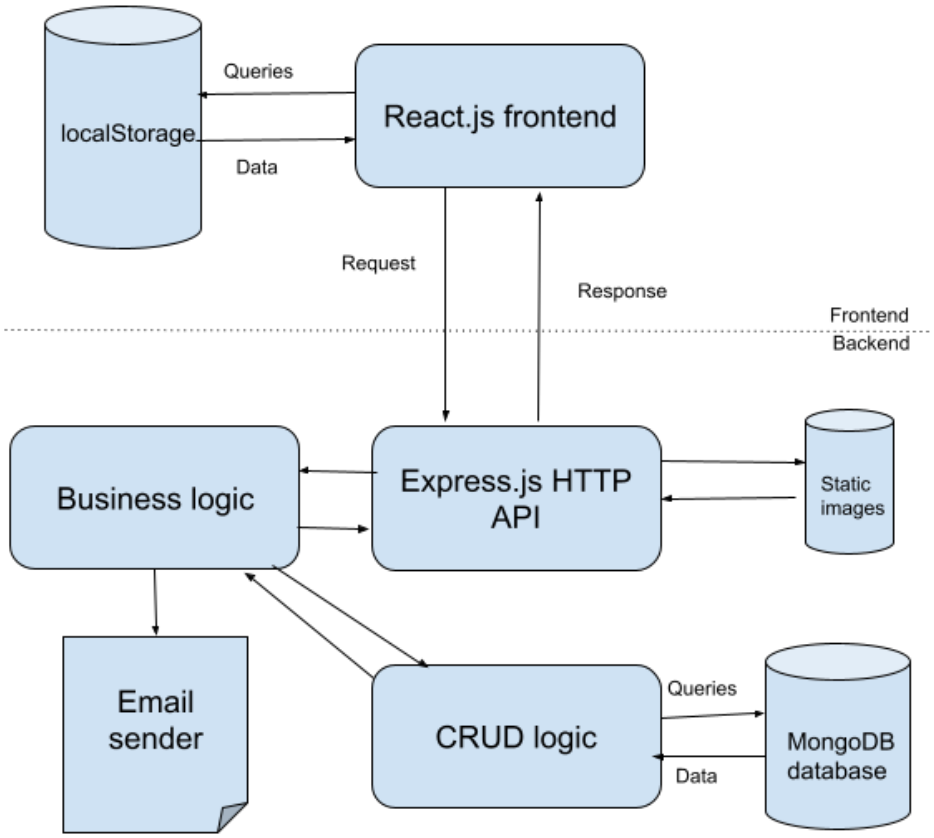
Criteria B: Design

All diagrams were made using Google Drawings
All design ideas were made using Mockflow
ER diagram was made using "Visual Paradigm Online" web application

Table of Contents

Overall Data Flow 2
UI flow 3
Backend data flow 5
Design mockups 9
Font and color scheme used 11
List of Javascript modules and functions with their purpose 12
Entity Relationship Diagram 14
Sample database records 15
Test plan 16

Overall Data Flow



Data stores:

- *localStorage:* Key-value database stored in the browser, which will be used for storing non-volatile data in the client (passkey and building name)
- Static images: Student images stored inside a folder on the server
- MongoDB database: Stores student and late arrival information

Layers:

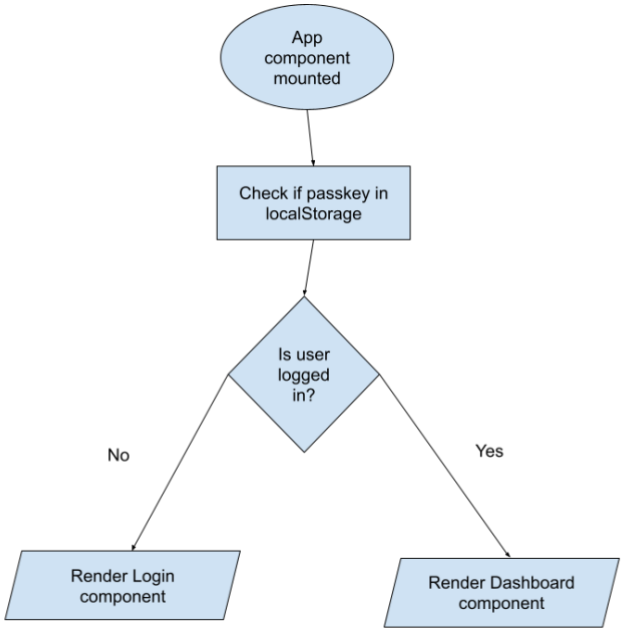
- Frontend layer: Contains the UI and the UI logic responsible for switching screens, displaying information, etc.
- API layer: Contains logic for processing requests and sending responses as well as request body validation
- Business logic layer: Handes logic specific to the solution such as detecting whether a student has been late thrice or formatting emails to be sent
- CRUD logic layer: Contains the logic for sending requests to the database to read, write or delete data.

Each layer will likely be implemented in a different file (for the backend). This layered architecture with each layer being abstracted from the others will allow the codebase to be more readable and maintainable.

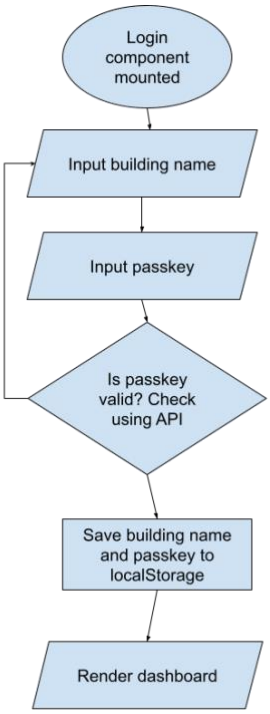
UI flow

These diagrams show the data and logic flow for each component in the UI (frontend) layer.

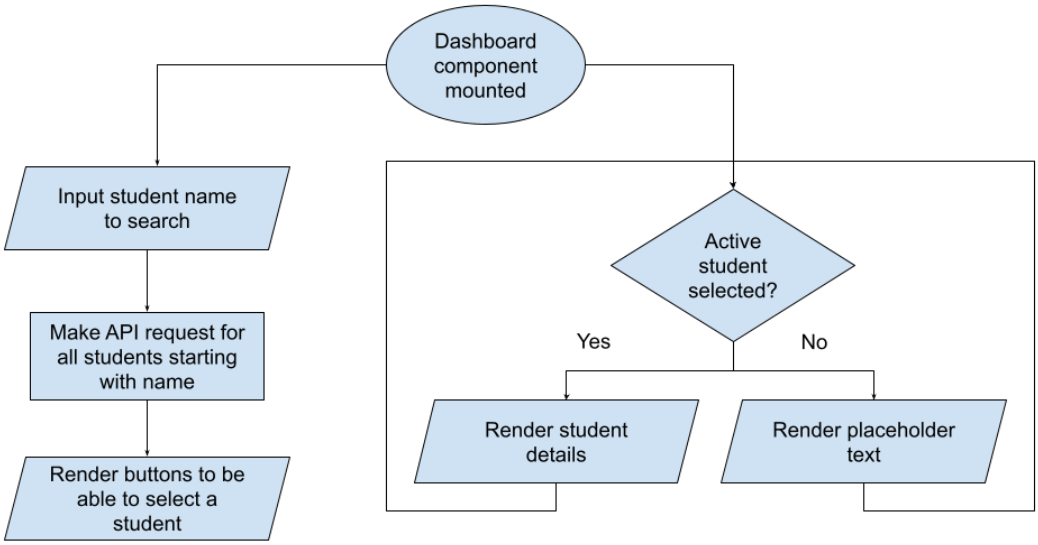
Main application



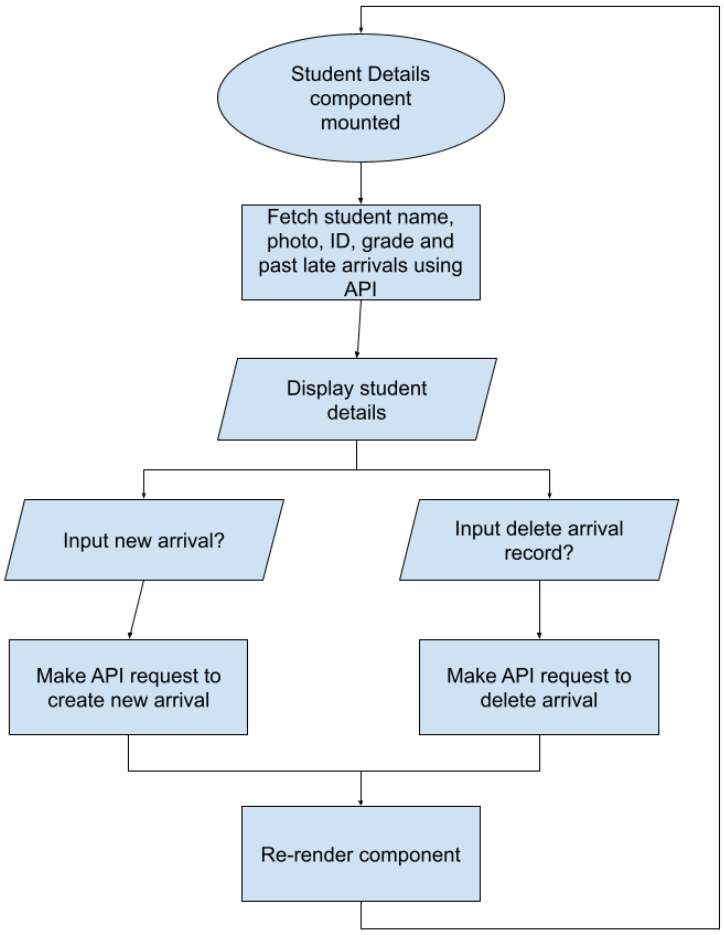
Login



Dashboard



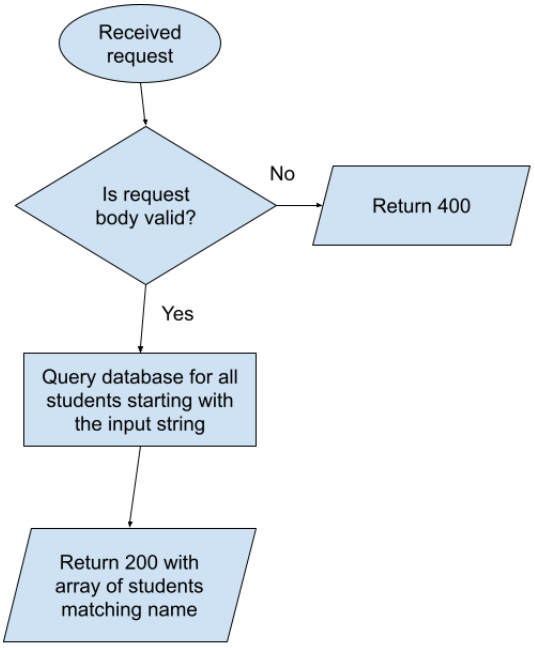
Student details



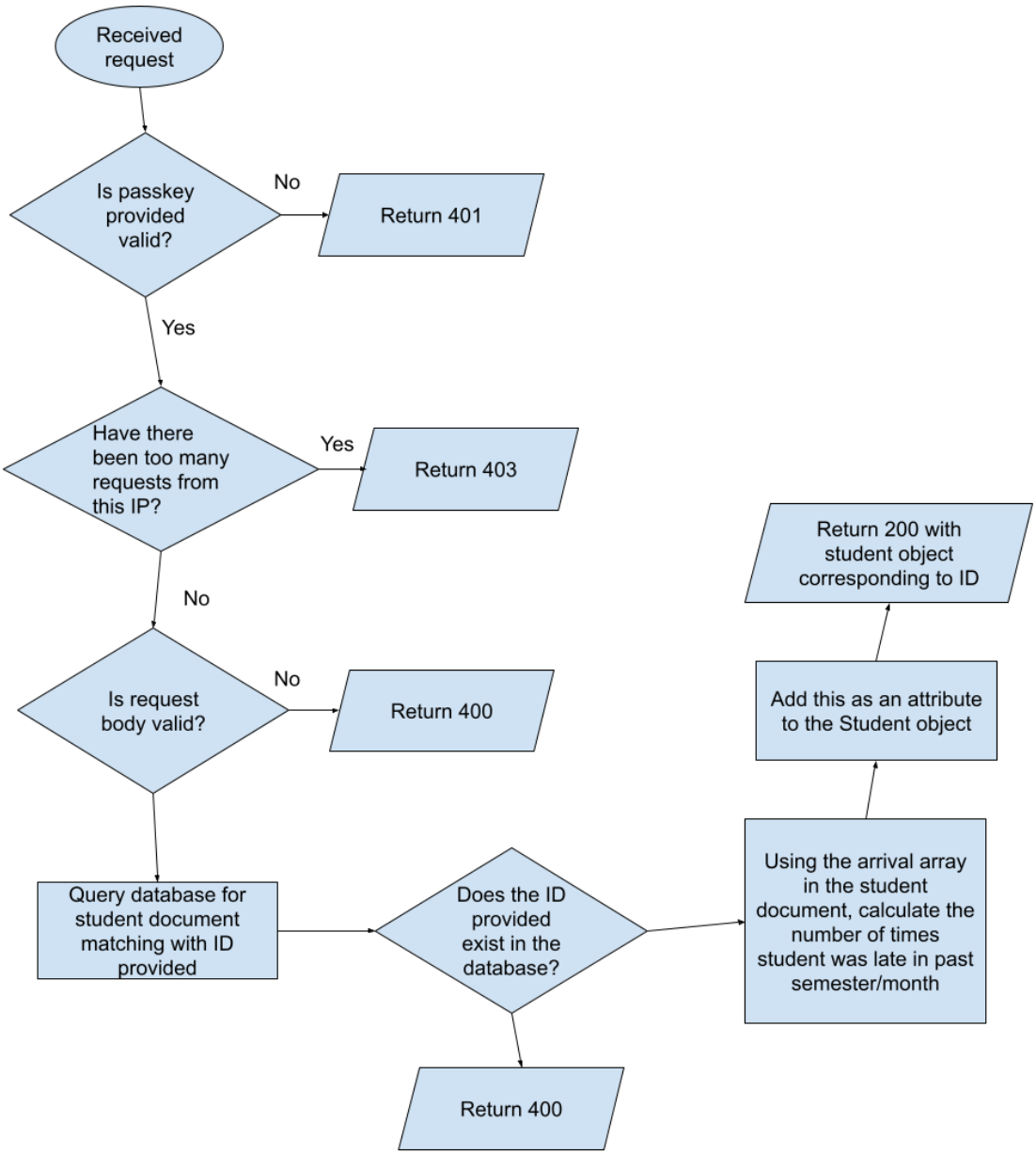
Backend data flow

These diagrams show the data flow for the API, business logic and CRUD logic layers (backend) for each API endpoint.

GET /queryName API endpoint
Input: Name
Output: Array of students with their names and IDs

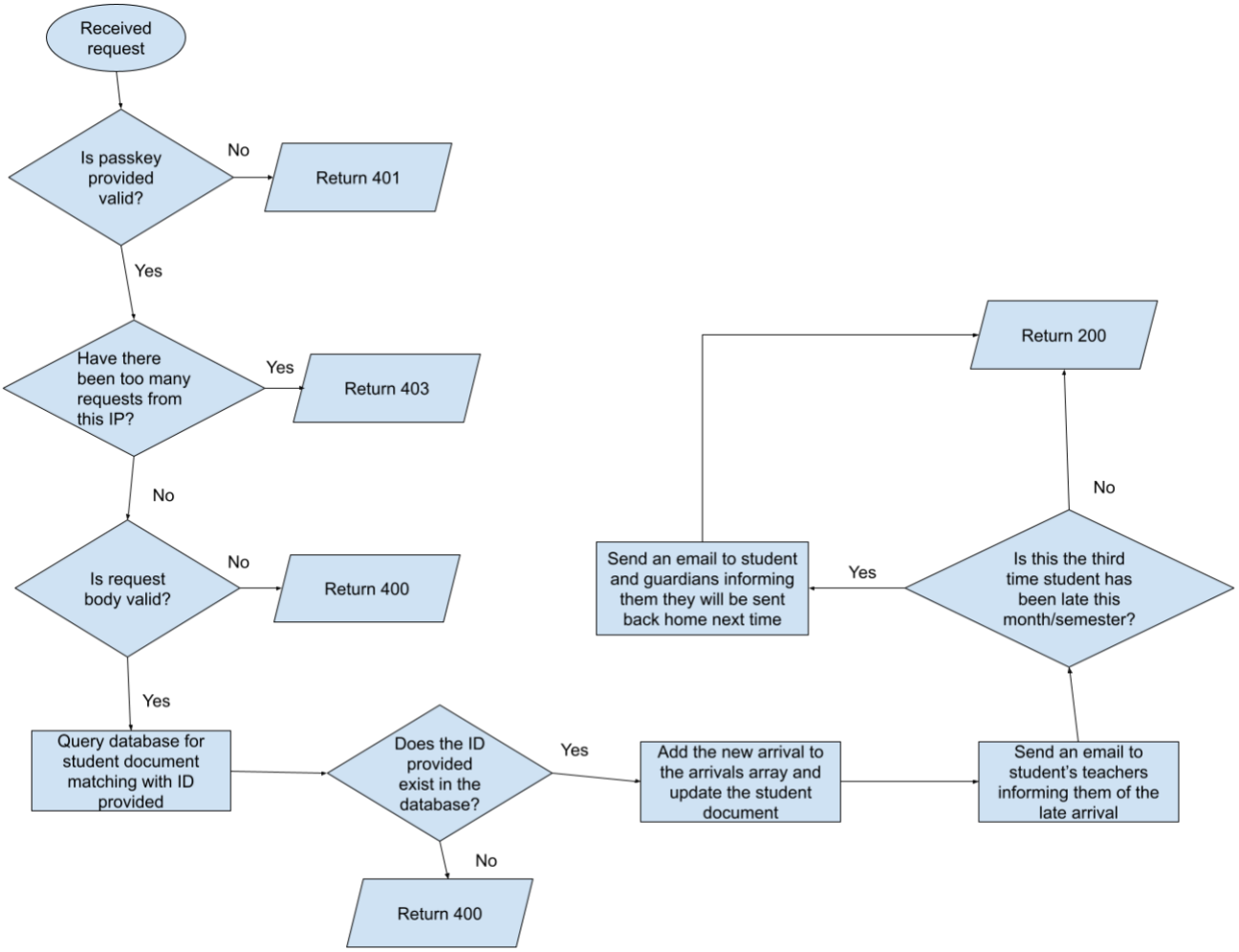


GET /queryID API endpoint
Input: Student ID, authentication (passkey)
Output: Student object corresponding with ID

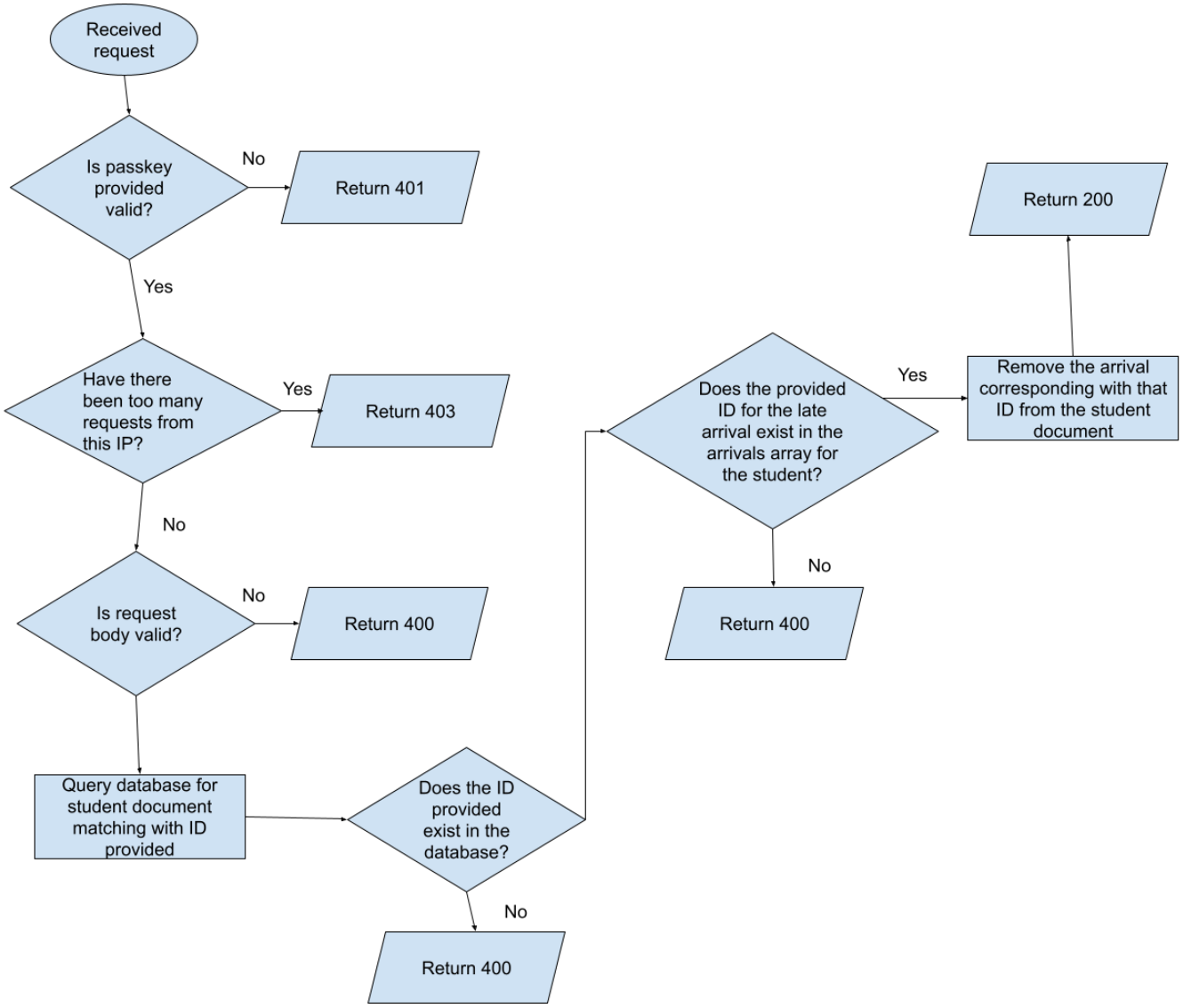


POST /arrival API endpoint

Input: Student ID, building name, reason for late arrival

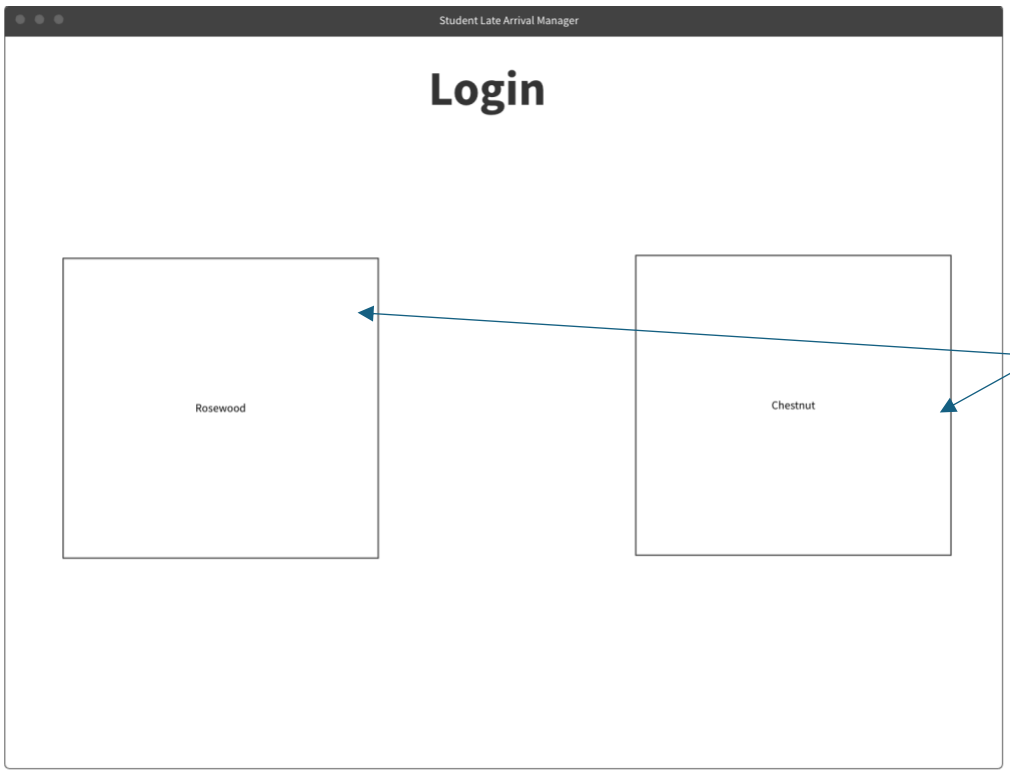


DELETE /arrival API endpoint
Input: Student ID, Arrival ID

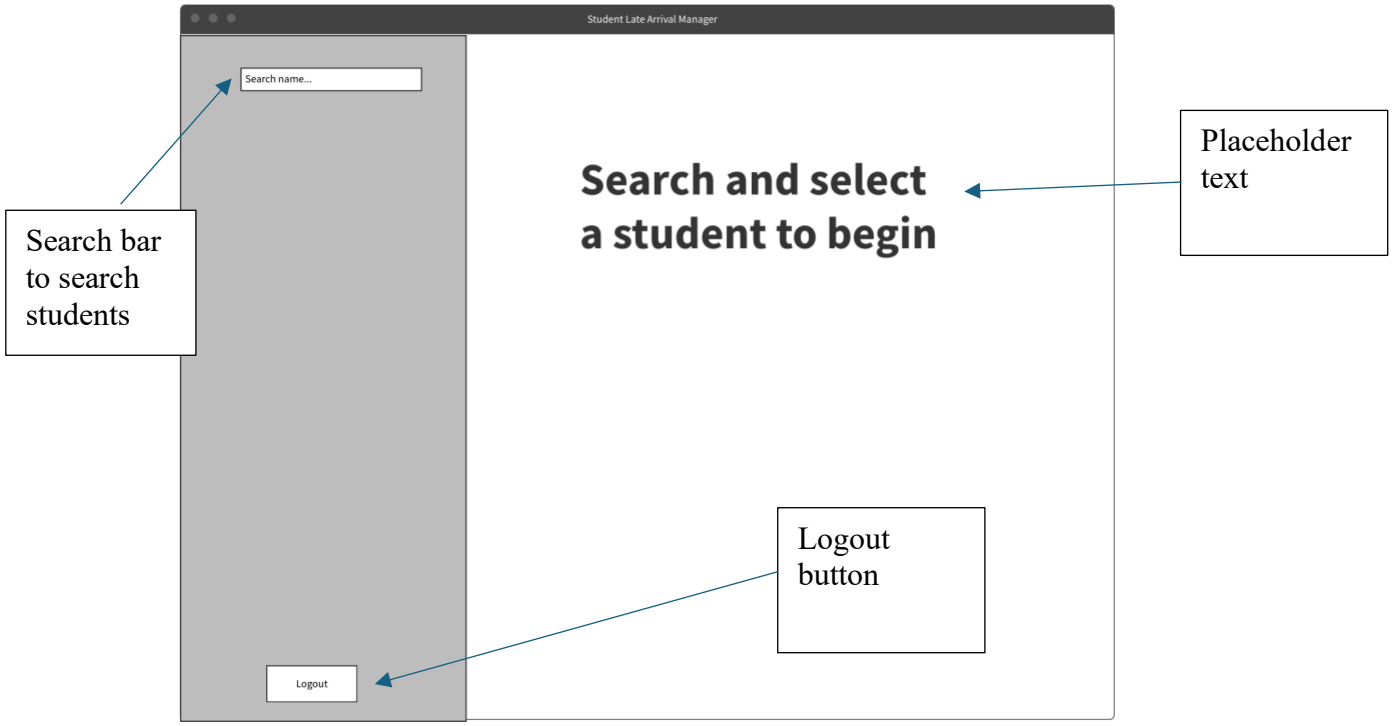


Design mockups

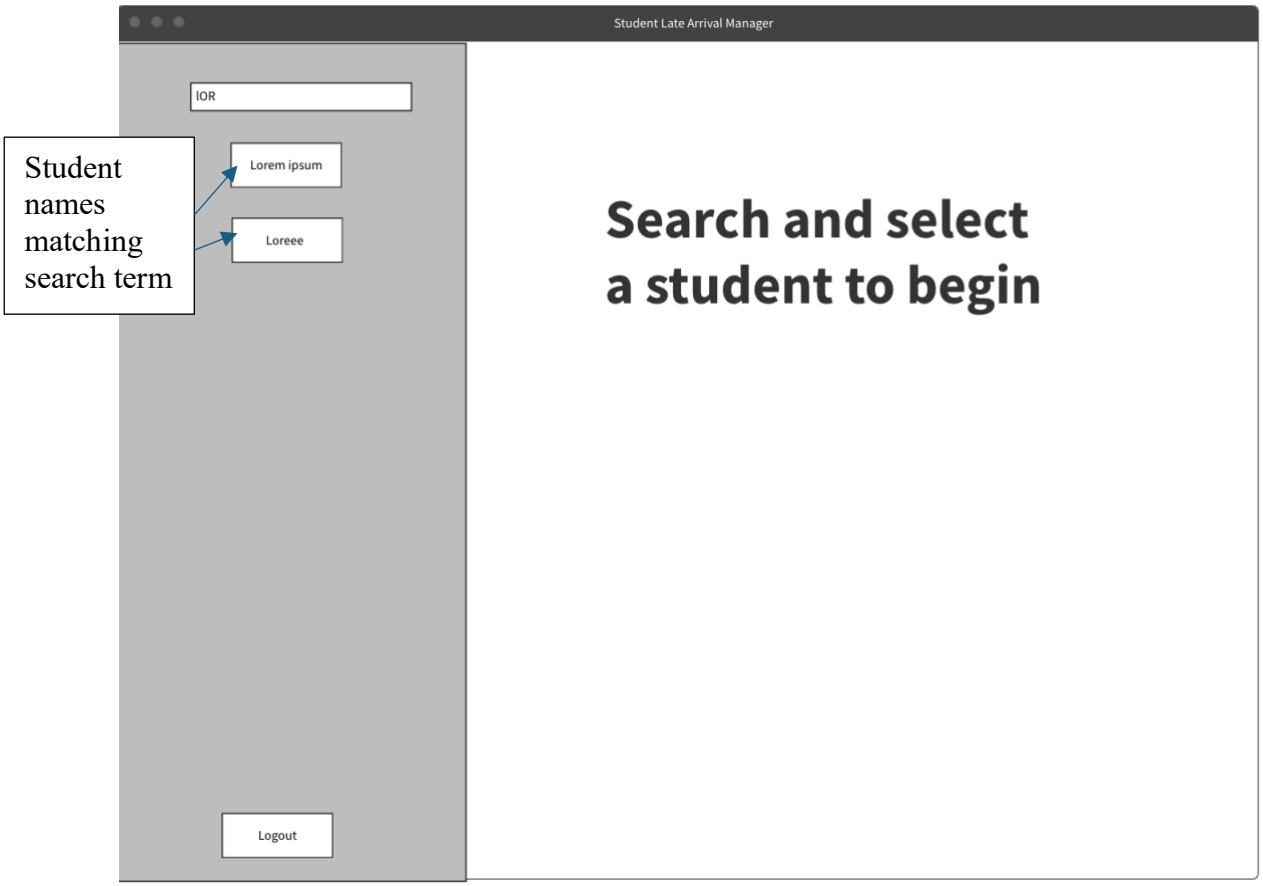
Login page



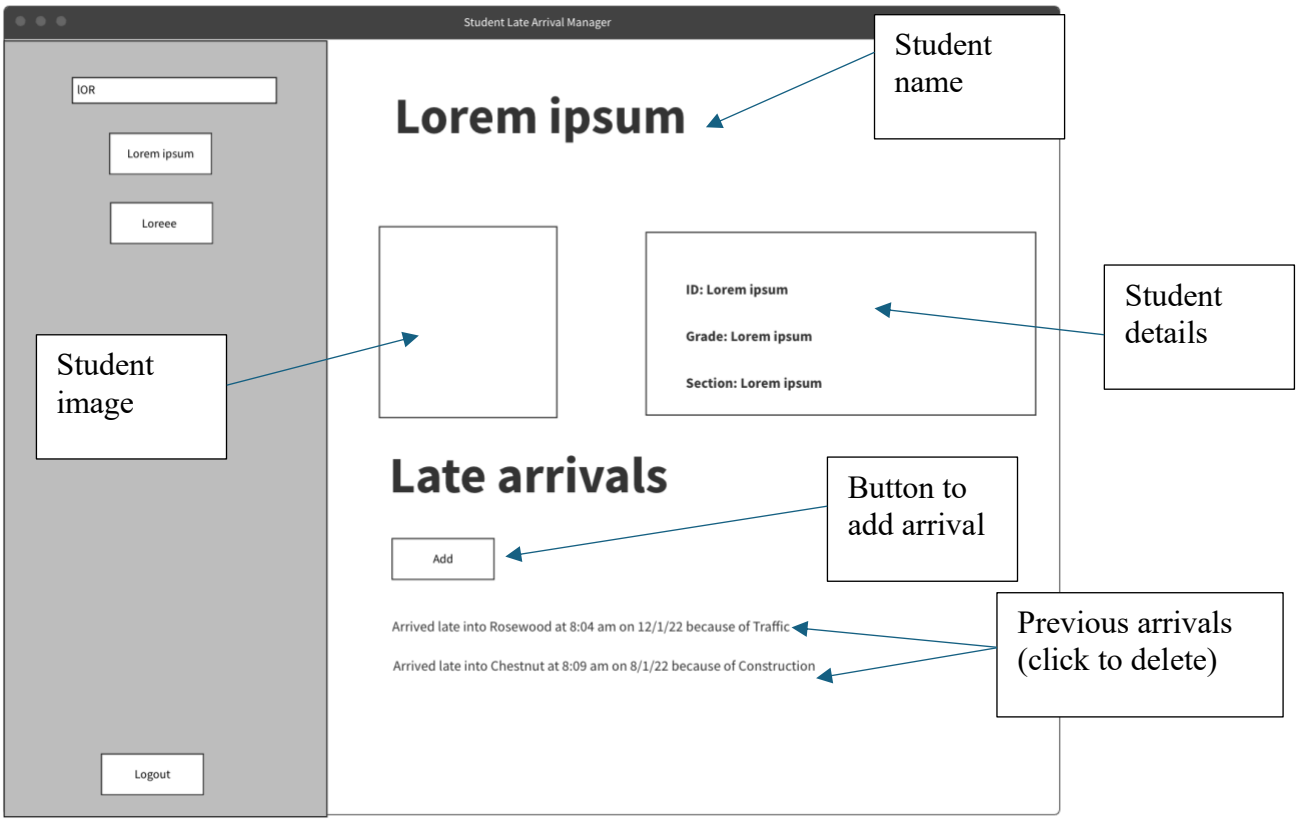
Dashboard initially



Dashboard after search

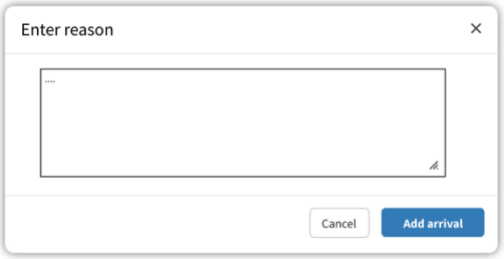


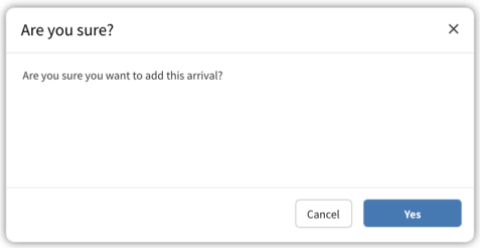
Dashboard after student selected



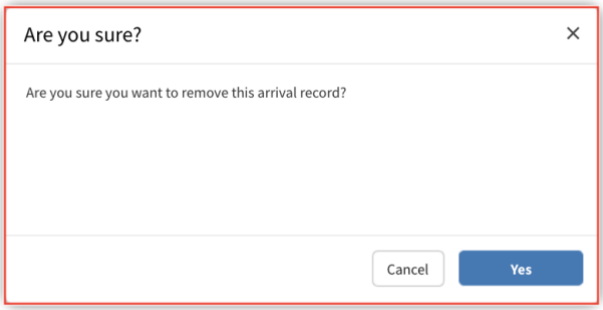
Dialogue boxes

Adding arrival (second box added after feedback from client – see Appendix B.1)

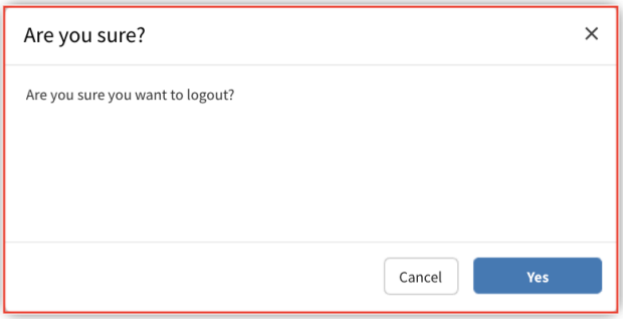




Removing arrival (added after feedback from client)



Logging out (added after feedback from client)



Font and color scheme used

Font: Montserrat

Colors:

Bootstrap standard colours for UI elements

(RGB)
229, 57, 45: Red for banner
212, 213, 214: Dark grey for certain backgrounds
238, 238, 238: Light gray for most backgrounds

List of Javascript modules and functions with their purpose

Backend

Module name	Export? (public)	Function name, arguments and return type	Purpose
index.js	-	-	Contains the routes and middleware for the API.
emails.js: Logic and template for sending the emails.	No	formatAMPM(Date): String	Convert date object into a string such as “11:23am”
	No	parseDate(String): Date	Convert dd/mm/yy string to date obejct
	No	sendMiddleSchool(String name, String email, String subject, String content): Boolean	Send an email from the middle school office
	No	sendSeniorSchool(String name, String email, String subject, String content): Boolean	Send an email from the senior school office
	No	isSeniorSchool(Student): Boolean	Determine whether a student is in senior school
	Yes	numTimesLate(Student): Integer	Calculate the number of times a student has been late within the appropriate window (month/semester)
	Yes	sendTeachers(Student): Boolean	Send an email to the teachers of a student informing them that they have arrived late
	Yes	sendGuardiansAndStudent (Student): Boolean	Send an email to the guardians and student (only run on the 3rd late arrival)
crud.js: Handle logic for CRUD operations on database	Yes	addLateArrival(String studentID, String building, String reason): Boolean	Add a new late arrival to the database
	Yes	deleteArrival(String studentID, String arrivalID): Boolean	Delete the specified arrival of a specified student
	Yes	queryStudentsStartingWith Name(String name): Student[]	Return all the students that start with the specified name (case insenstitive)
	Yes	queryStudentByID(String id): Student	Return the student information of the student with the specified ID

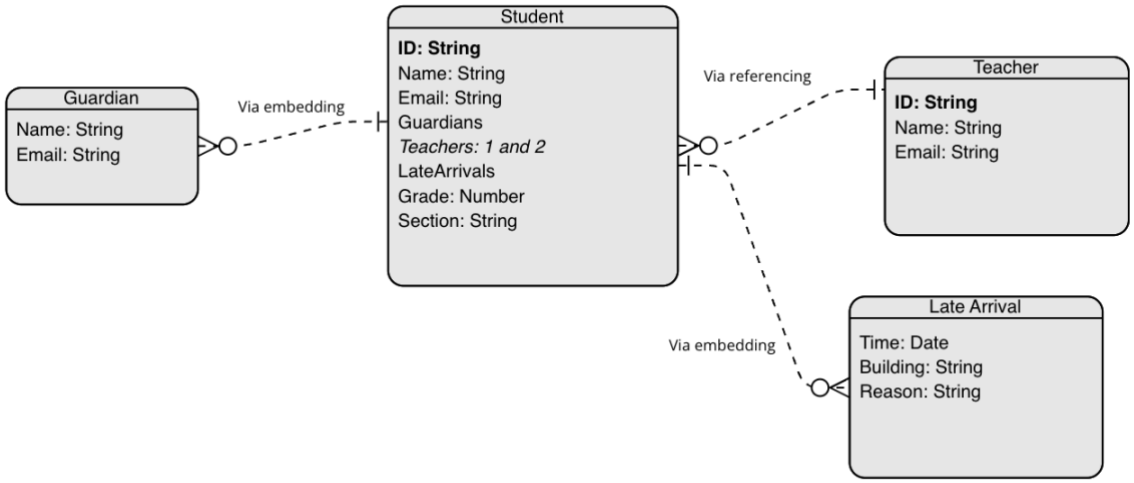
constants.js	Yes	-	Contains constants such as the start dates of each semester.
lateArrival.js	Yes	-	DB schema for late arrival collection.
LateArrival.js	Yes	-	DB model for late arrival collection.
Student.js	Yes	-	DB schema and model for student collection.
Teacher.js	Yes	-	DB schema and model for teacher collection.

Frontend

Module name	**Export? (public)**	**Function name, arguments and return type**	**Purpose**
index.js	Yes	-	Renders the root component.
App.js	Yes	App(): ReactElement	Root component that renders either the dashboard or login page
api.js: Helper module for API connections and authentication.	Yes	getImage(String id): String	Returns the path to an image of a student with specified ID
	Yes	buildAuthenticated()	Build an authenticated axios client using the passkey from localStorage
Dashboard.jsx	Yes	Dashboard(String buildingName, function onLogout): ReactElement	Renders the main dashboard of the application
StudentDetails.jsx	Yes	StudentDetails(String id, String building): ReactElement	Renders the student details for the provided student ID
	No	getData()	Update the student details using the data of the student fetched from the API using the ID
Warning.jsx	Yes	Warning(Integer timesLate, String timePeriod): ReactElement	Displays the warning that a student has been late a certain number of times the previous semester/month
LateArrival.jsx	Yes	LateArrival(Arrival arrival, function onDelete): ReactElement	Display a late arrival record with delete button

	No	formatAMPM(Date): String	Convert date object into a string such as “11:23am”
	No	formatDDMMYY(Date): String	Convert date object into a string such as “05/11/22”
Login.jsx	Yes	Login(function onLogin): ReactElement	Renders the login screen with the two building options.
BuildingSelector.jsx	Yes	BuildingSelector(function onLogin, String buildingName): ReactElement	Returns a button for the specified building name allowing the user to login to that specific building

Entity Relationship Diagram



To design the database structure, I made an ER diagram using an online tool. Although ER diagrams are traditionally meant for relational databases, I adapted it to work for the document-based database being used.

Sample database records

Student

```
{
  _id: ObjectId('65f73c040b0e8f543b26502e'),
  name: 'Alexander Brown',
  grade: 9,
  section: 'C',
  email: 'alexander.brown@example.com',
  guardians: [
    {
      name: 'Hugh Jackman',
      email: 'hugh@example.com',
      _id: ObjectId('65f73c040b0e8f543b26502f')
    },
    {
      name: 'Anne Hathaway',
      email: 'anne@example.com',
      _id: ObjectId('65f73c040b0e8f543b265030')
    }
  ],
  teachers: [
    ObjectId('65f73c040b0e8f543b265026'),
    ObjectId('65f73c040b0e8f543b265028')
  ],
  id: 'PSN67890',
  lateArrivals: [],
  __v: 0
},
```

Late Arrival

```
{
  arrivalTime: ISODate('2024-03-18T04:23:19.857Z'),
  building: 'Chestnut',
  reason: 'traffic',
  _id: ObjectId('65f7c1b7497e46bc01b482cb')
},
```

Teacher

```
{
  _id: ObjectId('65f73c030b0e8f543b264ff4'),
  name: 'Barack Zoebama',
  email: 'barack.zoebama@gmail.com',
  __v: 0
}
```

Test plan

Type	Input/Action/Situation	Expected output/result	Success criteria
Normal	User tries to login using a correct passkey after clicking a building button	User is told that the login successful, building and passkey are stored in localStorage	1, 14
Normal	User tries to login using an incorrect passkey after clicking a building button	Login unsuccessful, user is told that the passkey is invalid	1, 14
Abnormal	User does not enter a passkey when prompted for one	Error message telling user to enter passkey	16
Normal	User enters some text in the student name search bar	All students in the database starting with the name (ignoring case) are displayed in the sidebar	2
Abnormal	User enters some text with special characters part of Regex syntax in the student name search bar	Special characters are escaped and do not interfere with the regex.	16
Normal	User clicks on a student displayed in the search results	The correct details of the student from the database (ID, grade, etc.) are shown in the dashboard	2
Normal	User clicks on a student displayed in the search results	All the previous late arrivals of this student stored in the database are displayed with the correct date, time, reason and building on the dashboard	3
Normal	User clicks on a senior school student that has had >=3 late arrivals in the current semester	Warning displayed that student has already been late too many times	4
Normal	User clicks on a middle school student that has had >=3 late arrivals in the current month	Warning displayed that student has already been late too many times	4
Normal	User clicks on a senior school student that has had <3 late arrivals in the current semester	No warning displayed	4
Normal	User clicks on a middle school student that has had <3 late arrivals in the current month	No warning displayed	4
Abnormal	User clicks on “Add” button for a selected	Error message telling user to enter a reason for being tardy	16

	student and does not enter a reason or clicks cancel		
Normal	User clicks on “Add” button for the selected student and enters a valid reason	A new late arrival record is appended to the user document for the selected user in the database. This arrival record has the current time, the building stored in localStorage and the reason entered by the user. User is informed that the arrival is added successfully. UI is updated to show the new record on the dashboard.	5, 6
Normal	User clicks on “Add” button for the selected student and enters a valid reason	Email is sent to the address and names of the form room teachers of student stored in the database with the reason and time of arrival.	7
Normal	User clicks on “Add” button for the selected student and enters a valid reason. The selected student is in middle school and has exactly 2 late arrivals for the current month.	An email is sent both to student and student’s guardians using the email addresses and names stored in the DB, informing that the student will be sent back home next time they are late	8
Normal	User clicks on “Add” button for the selected student and enters a valid reason. The selected student is in middle school and does not have 2 late arrivals for the current month.	No email is sent to student or students guardians.	8
Normal	User clicks on “Add” button for the selected student and enters a valid reason. The selected student is in senior school and has exactly 2 late arrivals for the current semester.	An email is sent both to student and student’s guardians using the email addresses and names stored in the DB, informing that the student will be sent back home next time they are late	8
Normal	User clicks on “Add” button for the selected student and enters a valid reason. The selected student is in senior school and does not have 2 late arrivals for the current semester.	No email is sent to student or students guardians.	8

Normal	User clicks on Remove button to remove a late arrival record	The record is removed from the arrivals array in the database. User is informed that the record has been deleted. UI is updated to show the late arrivals without the deleted arrival.	9
Normal	Logout button is clicked	Login screen is displayed and the passkey is removed from localStorage	15
Normal	Data import script is run using a json file with sample data in the same format as Appendix A.3	The sample data is correctly imported into the database (no data missing or corrupted)	10
Abnormal	Data import script is run using a json file with invalid sample data (invalid email IDs, sections, etc.) in the same format as Appendix A.3	An error is thrown in the script and data is not imported into the database.	16
Normal	Time is midnight	Cron script automatically backs up database into specified folder	11
Abnormal	Request is sent with a studentID or recordID that does not exist in the database (test with all API endpoints)	Server responds with 400 error	16
Abnormal	Sending requests without authentication (test with all API endpoints)	Server responds with 401 error	13
Abnormal	Sending requests with query or body not having the fields required (test with all API endpoints)	Server responds with 400 error	16
Abnormal	More than 50 requests sent from the same IP within 15 minutes (test with all API endpoints)	Server responds with 429 error	14