Technical Design Specification

Team Brogrammers

Overview

Plannit is an application that engages people of all backgrounds in an effort to give them the opportunity to make their days easier through **organization** and **management.**

Plannit provides its users with **modules** that will be presented in different categories that the user specifies on his or her homepage.

With Plannit, the user could have something as small as a **calendar** to put down important dates, or could go as big as a **reminder system**, your entire **notes** for a class, and even more!

Team

Jake Bernstein

Design Manager / Full-stack developer

Matthew Lydigsen

Project Manager / Back-end developer

Lucas Meira

Database management / Back-end developer

Michael Markman

Financial guru / Full-stack developer

Shaoxi Ma

Head UI designer / Front-end developer

Raymond Zhu

Database Integration / Back-end developer

Why MEAN Stack?

For this app we will be using the MEAN stack to do all of our coding. The MEAN stack uses four pieces of software: MongoDB, ExpressJS, AngularJS, and NodeJS. Using these four tools together will let us create an efficient, well organized, and interactive application quickly.

The MEAN stack benefits greatly from the strengths of Node. Node let's us build real-time open APIs that we can consume and use with our frontend Angular code. Transferring data for applications that require quick display of real-time data. Plannit will be very front-end heavy and very reliant on displaying our modules in a timely fashion, so this will be very useful.

Why MEAN Stack?(Cont.)

Since every component of the stack uses JavaScript, we can glide through our web development code seamlessly. Using all JavaScript lets us do some great things like:

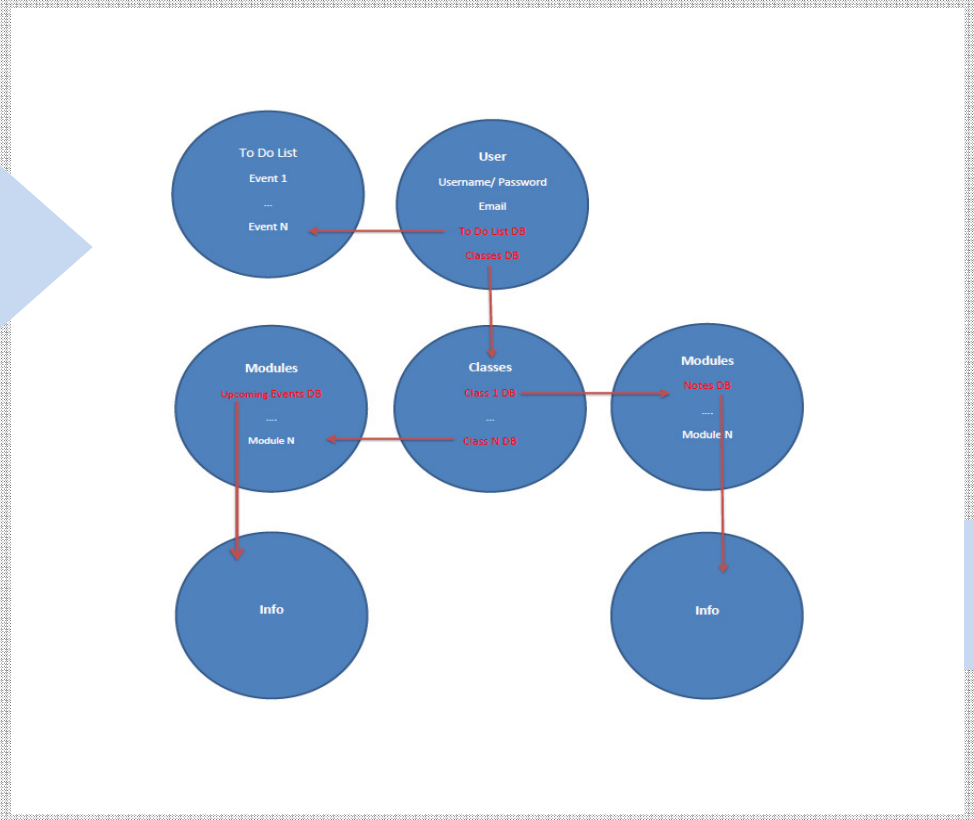
- Use JavaScript on the server-side (Node and Express)
- Use JavaScript on the client-side (Angular)
- Store JSON objects in MongoDB
- Use JSON objects to transfer data easily from database to server to client
- A single language across your entire stack increases productivity. Even client side developers that work in Angular can easily understand most of the code on the server side.

MongoDB

MongoDB is an easy to use database framework that provides high performance, scalability and availability.

For our application, lib/user.js will initialize MongoDB and get the databases of all users. Every user will have a username, password, email, and a uid. In addition there will be a default database for the “To Do List” and a database used to store that users’ course names. For each course, there will be a designated database for that specific class. These course database will store the modules needed for that respective course. And depending on the module, for example “Notes” and “Upcoming Events”, there will also database for those modules with a list of information.

Mongo Bird's Eye View



MongoDB Component Breakdown

exports.addNewUser = function(username, password, email) { … }

- This will add a user to the users database and ensure that this user isn't already an existing user
- exports.removeUser = function(username, uid) {...}
- This will remove the user from the database
- exports.addHomeModule = function(user, nameOfModule) {...}
- This function will be called to add a unique database that will store all of the users' planners (for example cs326, cs250, etc.)
- exports.removeHomeModule = function(user, nameOfModule) {...}
- This function will remove a planner from the users database of planners

MongoDB Component Breakdown Part 2

exports.addPageModule = function (user, nameOfModule, newPageModule, pageModuleData) {...}

- This function is used in order to create a new, unique page module database that is specific to a user and one of their planners.
- The newPageModule variable will be the name of the new database
- The pageModuleData will be the data you store in that newPageModule, such as notes you may have or possibly a class link.
- exports.editPageModule = function(user, nameOfModule, pageModule, pageModuleData) {...}
- This function is used to edit a page modules data
- exports.removePageModule = function(user, nameOfModule, pageModule) {...}
- This function removes a specified page module and all data associated with it

Express

Node.js is a platform that provides a free environment to develop server side functions. Express is the web application development framework that we are using to build Plannit based on Node.js platform.

Routing is a main speciality that Express provides us to perform the communication between server side and client side. Routers can handle the request from the client, get the data from database, then send the response back to the client.

Express Bird's Eye View

app.js is our main entrance to our application on the server side. Thus, it is a core component that builds the base for our app. It is very important because it sets up the HTTP server that allows users to connect to, handles some connection errors for routers, then call routers to process the request according to the routes.

We will use most of the default modules that Express generates for us, such as:

- express (Express basic functions)
- path (path manipulation)
- serve-favicon (Plannit logo)
- morgan (logger)
- cookie-parser (handling cookies)
- body-parser (parsing different type of data)

Angular

Angular.js is a Javascript framework that was built by google to provide fast and dynamic front-end deployment. Angular allows you to build normal HTML applications and then extend the markup to create dynamic components. Angular has two main features that will assist in the creation of Plannit. These features are data binding, which deals with how we handle data in our application, and dependency injection, which deals with how we architect our data.

Our app Plannit will require a lot of different modules, functions, and user storage, which would normally require us to store our data in a lot of different locations. This would end up making the coding hard and finding the right data even harder. But Angular keeps this simple for us. Angular has something similar to a model view controller that keeps all the data in one spot so it is easy to find and edit!

Angular bird's eye view

Below is a list of all of our ejs files which will handle the html/css side of the code. Our routes files users.js and login.js (in node section) will render these files at specific routes and then when the client has to send data back to the server these files will go to a different route in either login.js or user.js which will then decide what to do with that data.

- addUser.ejs
- headerLogin.ejs
- home.ejs
- module.ejs
- login.ejs
- headerUser-ModulePage.ejs

Angular Component Breakdown

- Our login.ejs file will display the login view which is where everyone starts before they are logged in
- addUser.ejs will provide a view to add new users. Once a user has entered all of their information the form will reroute to their new homepage and enter their data into our users database
- HeaderLogin.ejs and headerUser-ModulePage.ejs will be our two default header classes. headerLogin.ejs will be the header for the login and addUser pages, which will not have a logout button since the user isn't logged in. headerUser-ModulePage.ejs will be the header we use for every other page on the site. This header will include a logout button

Angular Component Breakdown Part 2

module.ejs will be our last view and it will be the view for all planners and modules associated with those planners. Here is where the user gets to customize their planners and add from our list of default modules whatever they want whether it be our notes module or our new budget module.

Node

Node is a javascript platform that is perfect for building network applications easily and quickly. Node helps us run all of our javascript code so in this section I will list out our routes files and how they will integrate our database code with our html code as well as how it interacts with our databases.

Node bird's eye view

Classes involved are users.js and login.js which are in our routes folder

- These two classes provide all of the routes that we will use to navigate through our website.
- login.js will be our home page and access to all user pages is restricted until the user is logged in.
- users.js will provide all routes for the users individual page based off of the users database.

Node Component Breakdown Login

login.js will include only a few routes which include /login, /login/newUser, and login/addNewUser. login/addNewUser is only used to actually add the new user to the database and then redirect to their new home page

- /login will be the default screen that everyone starts in and a user can only go to this page and /login/newUser page until they are signed in
- /login/newUser will have a simple form that you fill out and then it will redirect to /addNewUser to actually add you as a new user and verify you are new user

Node Component Breakdown Users

users.js will include a few more routes but still not too many overall.

- /users/home will be the home page for all of our users and will display all of their planners they have created and their to-do list
- /users/addHomeModule, /users/removeHomeModule, and /users/editHomeModule will edit the current planners listed on the users home page
- /users/editToDoList will modify the to-do list on the users home page
- /users/logout will logout the user

Node Component Breakdown Users Part 2

- /users/nameOfHomeModule this path will take you to a page that contains one of your planners. The path name nameOfHomeModule will be replaced with the specific planners title. On this page we will put every module that the user has added to the page in this view. This is the part of the website where users can truly build their planner the way they want to.
- /users/nameOfHomeModule/addPageModule will add a module to the planner you are in depending on which module you specify
- /users/nameOfHomeModule/removePageModule will remove the specified module
- /users/nameOfHomeModule/editPageModule will edit the module you specify possibly the notes section or one of the other modules

Revision

Date	Author	Ver.	Changes
03/05	Matthew Lydigsen	1.0	Logo for website created
03/06 – 03/08	Jake Bernstein, Matthew Lydigsen	1.0	Initial layout of website created
03/07	Raymond Zhu, Shaoxi Ma	1.0	Default modules are decided
04/01	Jake Bernstein	2.0	Budget module added to list of default modules.
04/05	Matthew Lydigsen	2.0	Layout of all parts of code for express app created.