UrScheduler

Team Alpha

Connor Grehlinger, Dongyao Lei, Ouhan You, Xingyu Lu

In modern days, people are spending more and more time in different meetings in both formal and casual settings. However, the process of setting up a meeting via phone call or email is usually time-consuming and inefficient. That's where our web application, UrScheduler, comes into play. Using the UrScheduler web application, meetings can be easily set up by simply clicking a link.

User Story

Steven is a computer science masters student attending Duke University in the hopes of finding a fulfilling and well-paying job. Steven knows that the first step to accomplishing this outside of his school work is to land a good internship and get professional experience. Steven is in the process of talking to recruiters at several companies which have selected him for an interview. However, Steven is very stressed and frustrated as finding an interview time at each of these companies that works for both him and the recruiter has been difficult. Setting up the interviews has involved a lot of emailing back and forth, and sometimes the interview time is still undecided even after multiple emails.

One afternoon while Steven is venting his frustration to a classmate named Jeff, Jeff tells him that his experience finding an interview time has been stress-free. This is because Jeff set up all his meetings with the UrScheduler web application. The UrScheduler web application offers a simple and coordinated way for two or more individuals to set up any type of meeting, not just interviews!

Steven takes his classmate Jeff's advice and suggests to some of the recruiters he is still talking to that they can more easily find an agreeable time with the UrScheduler web app. The recruiters are reluctant to use the UrScheduler web app at first as they have never used it before. But Steven explains to them that using the UrScheduler is simple, and all they have to do is wait for an email sent to them containing a link to set up an interview or meeting time.

Steven signs up for the UrScheduler service through the web application's easy sign-up service. Steven then navigates to a page which lets him create a set of dates and times which match his availability. Each of the recruiters receives an email with a link that takes them to the UrScheduler website. After each recruiter registers with the UrScheduler web app, a page loads which shows the set of dates and times in which Steven is available. The recruiters are each happy as all they have to do on this page is select a date and time from the set that also works for them. The selected time is then added to Steven's and the recruiter's calendar, and removed from the set of available times the rest of the recruiters can select from. Steven is relieved of his stress as he has now set up all of his interviews without the hassle of emailing back and forth and coordinating with other recruiters. Some of the recruiters are so pleased with the UrScheduler's performance they decide to use the service to coordinate with other interviewees.

Product Requirement

Functional Requirement

1. Users should be able to create a personal account that allows them to use the UrScheduler web app.
   1.1. This account should bind with the user's email and be password protected.
2. Users should be able to create and configure meetings.
   2.1. A meeting can be either a one-on-one meeting or a group meeting.
   2.2. Users can configure the name of the meeting, meeting date, time, duration, location, and maximum number of participants.
   2.3. The meetings can be edited after the event is created
3. For both meeting types, event creator and event participants can coordinate a time to meet.
   3.1. Event creator should have the ability to schedule their availability for meetings.
   3.2. Other participants should be able to view the event creator's availability and confirm a meeting time.
4. Users should be able to send and receive meeting requests for the UrScheduler via email.
   4.1. A request email will be sent to whom the user wants to meet with.
   4.2. Email requests will contain at least one link to redirect the recipient to the UrScheduler web app.
   4.3. Invitation receiver can choose time slot or refuse the invitation.
5. Users can create a team and invite people to join. Events can be created targeted at a team.
6. People who are not registered users of UrScheduler can also send and receive meeting invitations.
7. (nice to have) Synchronize to Google Calendar/Calendar in iOS and macOS

Non-functional Requirement

Platform

This is a web application which should be compatible with Google Chrome/Safari, etc. The website will work the same way when users visit it from a mobile browser.

Programming Language

The web application will be programmed in Java and use the Spring framework. Html/Javascript/CSS will be included for the frontend development.

Security

User privacy must be protected.

1. Password must be encrypted.
2. User availability can only be seen if the user chooses to allow it.

Ease of Use

This website is going to be user-friendly.

1. Users with no experience can be familiar with this application without instruction in less than 5 minutes.

Reliability

1. All the emails are guaranteed to be sent.
2. If one invitation requests fails (e.g. the slot already occupied but the page not refreshed), the requester should get the error message and updated information.

Release Date

This application is scheduled to be deployed on Heroku before May.

Architecture Model

Browser-based User Interface

Configuration Services

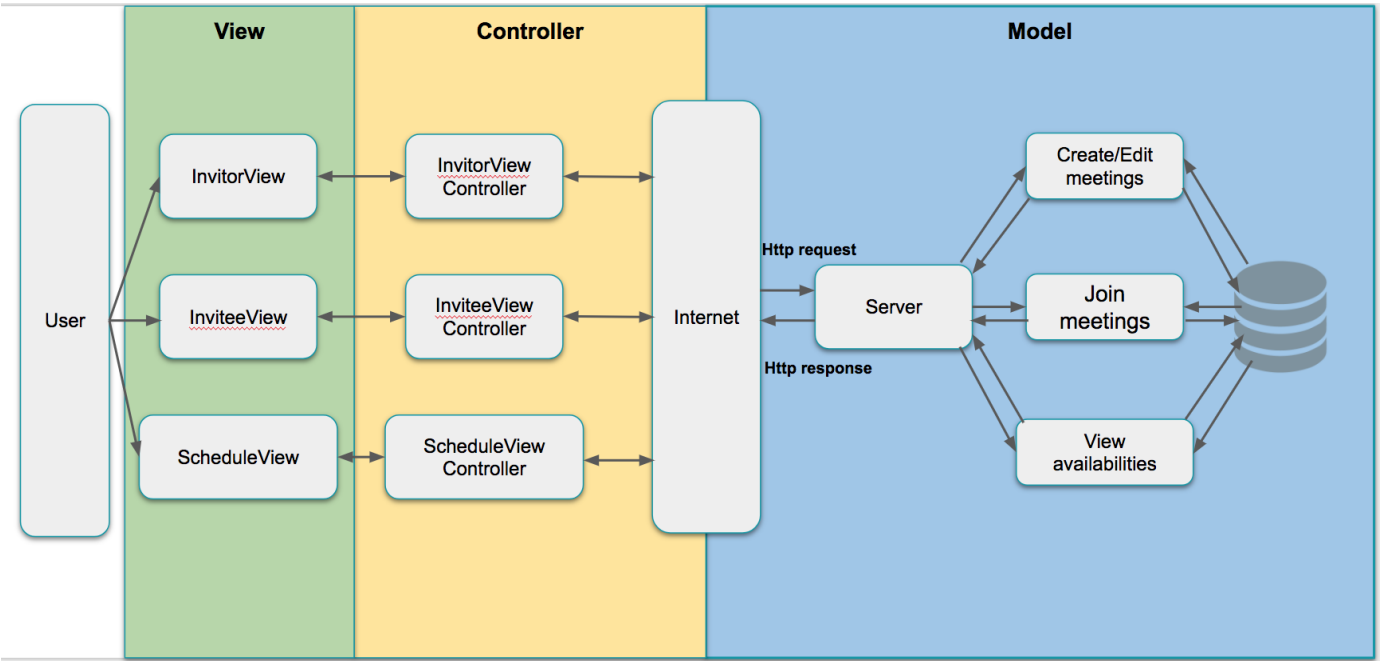
Meeting Management, Schedule Management, Group Management(Role Checking)

Application Services

Email notification, Meeting history archive, Calendar synchronization, Meeting registration, Meeting creation/edition, Massive invitation

Utility Services

Logging/authentication, User storage, Application storage



Use Case Model

Invitation Send	
Actors	Inviter, invitee, web server, database
Description	Inviter can pick several available periods on his calendar and send invitation to an invitee
Data	Inviter's available time, the kind of meetings
Stimulus	User can select one or more available time on his calendar then send meeting invitations by specifying other users or email addresses and clicking the button submit invitation button
Response	Web server will send an invitation email to the invitee. There will be a confirmation status icon which informs the user that invitation has been sent.

Response to Invitations	
Actors	Inviter, invitee, web server, database
Description	Invitee can either accept or reject the invitation. If the invitee accept the invitation, he should select a time from the provided time then the inviter and invitee will not be available at the selected time. Their other invitations will be dynamically updated, which means this period cannot be select in the other invitations of the invitee's and the inviter's.
Data	Inviter's available time, invitee's available time
Stimulus	Invitee can click on buttons to response the invitations
Response	There will be a confirmation status icon which informs the users that the meeting has been confirmed.

Meeting Set in Current List	
Actors	Inviter, invitee, web server, database
Description	Once invitee has selected a date and time that works for both them and the inviter, the meeting is marked in the inviter's list
Data	Inviter's meeting list, the kind of meeting, date and time for meeting
Stimulus	User will receive notification and see the invitees selected date and time for meeting
Response	Web server will submit response to notify inviter and invitee that the meeting date and time has been established. The web server saves the meeting information to the inviter and invitees meeting list (information saved in database)

Planning

Project Documentation Plan

Task ID	Description	Owner	Start	Finish	Dependencies
Rqmts	The initial requirements document needs to be cleaned up, refined, and finalized	Team Alpha	01/27	02/07	None
SysArch	The draft system architecture needs to be updated and finalized	Team Alpha	02/07	02/11	Rqmts
UseCase	The actions/event steps defining interactions between user and app needs to be designed.	Team Alpha	02/07	02/11	Rqmts
ProjPlan	The overall plan for both documentation and code development needs to be established	Team Alpha	02/15	02/18	Rqmts, SysArch
ModDsgn	The models which manage the data in the app need to be designed and well explained	Team Alpha	02/20	02/25	SysArch
README	The thorough explanation of the app, including both non-technical and technical aspects, needs to be clearly written	Team Alpha	04/10	04/15	Rqmts, SysArch, ModDsgn, UseCase

Project Plan

Sprint 1 (Feb.18th - Mar.4th)

Sprint 1 will be delivered on Mar 4th and the application should have scaffolding structure . Specifically, the application should be able to connect to the database and add entry to the database. It will also contain a template web page which holds static web components like head and navigation bar. Based on the template page, the application will have a simple dashboard.

Task ID	Description	Owner	Start	Finish	Dependencies
DbDsgn /Imp	Databases for managing users' personal information, meeting specification, meeting history, etc.	Team Alpha	02/18	02/23	None
Template	The base.html page which provides a template (including logo, navbar, etc.) for all its relative pages	O	02/18	02/23	None
Dashboard	Dashboard for presenting and editing user's meetings/settings/personal information	X	02/24	03/05	DbDsgn/Imp, Template

Sprint 2 (Mar.5th - Mar.27th)

Sprint 2 will be delivered on Mar 27th. The main work of sprint 2 focuses on presenting the current availability of certain meeting. Furthermore, after a meeting time is finalized, this time slot is no longer available for future participants. At the same time, authentication and group creation functionality will be added. Since these two tasks only depend on tasks in sprint 1, they can be developed at the same with calendar UI and calendar scheduling.

Task ID	Description	Owner	Start	Finish	Dependencies
Config Meeting Part1	Configuration of meeting specification (such as meeting type, duration,	C & D	02/24	03/08	DbDsgn/Imp, Template

	meeting name, link-generation) for registered users.				
Config Meeting Part2	The frontend of calendar for meeting creator to choose his/her availability. Also, store the availability information back to database.	X & O	02/24	03/08	DbDsgn/Imp, Template
Meeting Config Combine	Combine the meeting configuration part 1 & 2.	Team Alpha	03/09	03/10	Config Meeting Part1, Config Meeting Part2
Auth	The frontend and backend of authentication. This includes signup, login, logout functionalities and authentication verification for every page.	D	03/11	03/15	DbDsgn/Imp
Scheduling UI	The frontend UI for users to schedule a meeting. This includes CSS design and library package import	X & O	03/11	03/20	None
SetUp Meeting	The controller which receives requests of a meeting scheduling and updates the database	C	03/11	03/20	DbDsgn/Imp, Scheduling UI, ConfigMeeting
~~**Group Creation**~~	~~The frontend and backend of creating a group. Enable user to create his/her own group so that he/she can invite a group of people to his/her meeting just by one click. (Note: the group information can only be seen by~~	~~X & O~~	~~03/21~~	~~03/28~~	~~Auth~~

	~~the group creator)~~				
ResponseView	The frontend for users to response a meeting request. Users can either accept or reject an invitation.	D	04/03	04/09	SetUpMeeting
Response in Db	The backend of response meeting requests. Database will record information that whether the user accept an invitation or not.	D	04/09	04/15	ResponseView

Sprint 3(Mar.28th - Apr.15th)

Sprint 3 will complete the Meeting Manager and be delivered on Apr 15th. Functions which allow users to respond to invitations, reschedule or cancel meetings will be added. Homepage which present access to all the functions and a brief introduction will be provided for users.

Task ID	Description	Owner	Start	Finish	Dependencies
SendEmail	Emails will be sent to verify registration, make invitation and remind users of meetings.	C	03/28	04/03	GroupCreation, SetUpMeeting
Meeting Edition UI	This interface allows users to modify existing meetings. It provide access to change a meeting's date, location or to cancel it.	X	03/28	04/03	SetUpMeeting, Template
DbUpdate	After meetings have been rescheduled, the corresponding information in database will also be	X	04/03	04/09	SetUpMeeting, Meeting Edition UI

	updated and the members of meetings should be notified.				
CancelMeeting	If the user chooses to delete a meeting in the Meeting Edition UI, the database must update that change and members of the meeting must be notified	O	04/09	04/15	SetUpMeeting, Invitation
HomePage	The homepage (before login) will provide a basic introduction of Meeting Manager	O	04/03	04/08	Template
~~**ResponseView**~~	~~The frontend for users to response a meeting request. Users can either accept or reject an invitation.~~	~~D~~	~~04/03~~	~~04/09~~	~~SetUpMeeting~~
~~**Response in Db**~~	~~The backend of response meeting requests. Database will record information that whether the user accept an invitation or not.~~	~~D~~	~~04/09~~	~~04/15~~	~~ResponseView~~
Invitation	Users can send invitations to whom the user want to meet with. This can be done via email or app notification (or both).	C	04/03	04/09	SetUpMeeting, GroupCreation
Group Creation	The frontend and backend of creating a group. Enable user to create his/her own group so that he/she can invite a group of people to his/her	X & O	04/08	04/12	Auth

	meeting just by one click. (Note: the group information can only be seen by the group creator)				

Testing

Test Methodology

Test Process)

For each Task ID listed in the sprint tables below, we follow or nearly follow the software testing process from the course textbook, see Fig. 1. We will be designing test cases for each task from our project plan document, and comparing the results our web application generates with the desired or expected results for that task. Some of this will be done with JUnit testing in Idea, while other tests will take the form of Validation Testing and Defect Testings (see below for details).

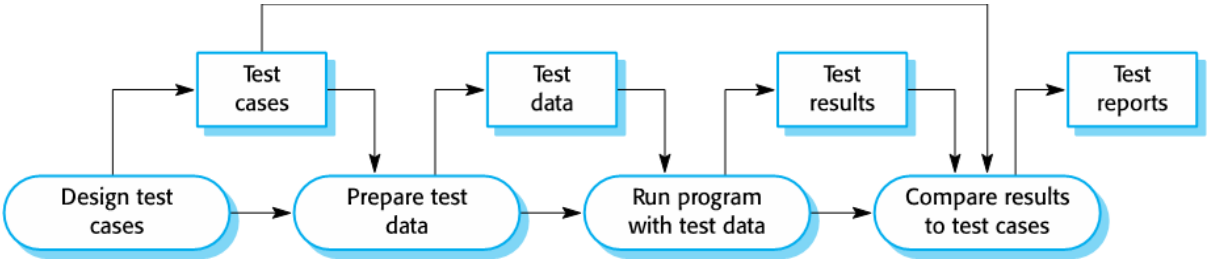


Figure 1 Software Testing Process Model

Test Stages)

Each sprint consists of the following three test stages:

1) Development testing
2) Release testing
3) User testing

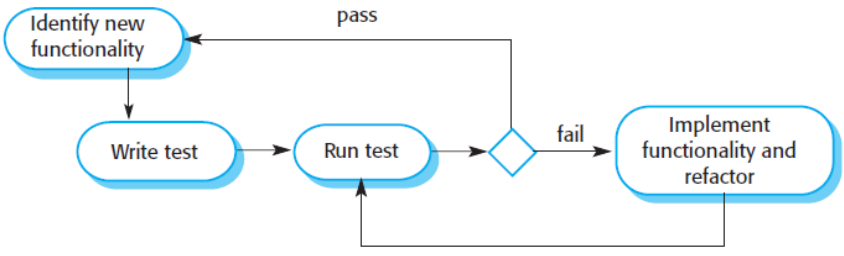


Figure 2 Test-driven Development

For the development testing, we will use Test-driven development(TDD). The process of TDD is shown in Fig.2. We start by identifying the increment of functionality that is required. This functionality should be small and implementable in a few lines of code. Then we will write test cases for this

functionality, write code, run and test it. If it failed the test case, we can easily check and rewrite it. After all test cases are passed, we can work on next chunk of functionality.

We will do release testing and user testing after each springs. At those time our application will have basic functions so we could imagine us as users and use our application. We will first perform as normal users and do some validation tests.Then we will perform as “naughty users” and do defect testing.

Test Methods)

In the development of our project, we will use three main testing methods:

1) Unit Testing - These test our code at the unit level and encompass both defect and validation testing. Each unit should be able to handle its typical input as well as incorrect or improperly formatted input and produce the desired output. We will carefully construct these test cases to try to include as many corner cases as we can identify in the three test stages.
2) Defect Testing - These tests are meant to identify any defects which may be present in our code. If a defect is discovered, we will patch it and re-test.
3) Validation Testing - These tests are meant to verify that our web application meets our project requirements.

Our component and system tests will mainly consist of defect and validation testing. This is to ensure that all components and the system as a whole work as expected and meet our project requirements. Typical or “real-world” input test cases, atypical input test cases, malformed input test cases, and test cases which stress the corner cases which arise in the use of our project will all be used. We will monitor these test cases for the desired outcome and check the functional steps our code takes which each.

Test Plan

Sprint 1 (Feb.18th - Mar.4th)

Task ID	Description	Sample Test Cases	Test Methods	Owner
Database Test	The basic operation with database (MongoDB) will be tested. We will use MongoDB Compass to test basic CRUD operations.	1. Insert invalid data into DB 2. Insert valid data into DB 3. Read/delete non-exist data in DB 4. Read/delete exist data	Defect Test Validation Test	Team
Navbar Test	Each web page that contains a navbar extends from the same html file called navbar.html. Since navbar allows user to redirect themselves between different pages, it must guarantee that the redirection succeeds in an expected way	1. Starting from Home page, click 'Upcoming Meetings' 2. Starting from 'Home' page, click 'Home' 3.Starting from 'Create Meeting' page, click 'Home,' then click 'History Meetings'	Defect Test Validation Test Unit Test	O
Template Test	Because our web application templates use a hierarchical structure, i.e. some web page inheritance from several parent web pages such as head.html, navbar.html. To make sure this hierarchy works, we plan to test from the top level so that the hidden hierarchy make no difference to the outside world.	1. Open the Homepage, and scrutinize the source code from the front end. 2. Open the Homepage, and test all the functionality in it. 3. Open the create meeting page, and scrutinize the source code from the front end. 4. Open the create meeting page , and test all the functionality in it. 5. For every web page that extend from other templates, following the same procedure.	Component Test Defect Test Validation Test	O

MainPage Test (Sprint 1 User/Release Test)	We will check if the main page is presented with meeting configuration stored in DB and templates & navbars. The data will be manually inserted via MongoDB Compass.	1. Direct to the main page to see if there is any error. 2. Check the data shown on the webpage. 3. Click each button in navbar and template.	Release Test User Test Validation Test	X

Sprint 2 (Mar.5th - Mar.27th)

Task ID	Description	Sample Test Cases	Test Methods	Owner
Configure Meeting Back-end Test	To enable user to create a meeting, the database needs to provide an interface to insert a document into the meeting prototypes collection. The first step to success in this task is to make sure the back-end interface works.	1. Run commands in MongoDB to insert a document into meeting prototypes collection. 2.Change the parameters of the inserted document to make sure that every possible document can be inserted correctly.	Unit Test Validation Test	C & D
Configure Meeting Back-End & Front-End Interface Test	After making sure that back-end works correctly, the interface between front-end and back-end needs to be tested as well. This is the second and last step to ensure the meeting document can be created successfully by user.	1. Using the creating meeting page, fill out the information of the meeting to be created, then click create button. Then check if it is shown under the upcoming event tab. 2.Using the creating meeting page, leave some required field blank and submit the form, see what happens.	Component Test Defect Test Validation Test	X & O
Registration Test	We will test if users registration is correctly implemented. The	1. Register with normal/valid information 2. Redundant register (identical user	Unit Test Validation Test Defect Test	O

	DB should be updated and the password needs to be encrypted.	information)		
Authentication Test	We will test if user can successfully log in if correct pair of account-password is provided. And any access to any webpage without login should be denied.	1. Use incorrect password to login 2. Use non-exist account to login 3. Access any web page without login 4. Login with correct information in DB	Validation Test Defect Test Unit Test	D
Meeting Setup Test	When the invitee of a meeting try to accept the invitation, he/she needs to choose the final meeting time from the current availability of inviter. In addition, if the invitee choose an unavailable time slot, the app should handle it gracefully.	1. Run commands in MongoDB to insert a document into meetings collection with the constraints that the specific meeting prototype must exist in the meeting prototypes collection. 2. Try the above test with an nonexistent meeting prototype, see whether it is handled properly 3. Use user interface to accept the invitation and check if the newly set up meeting shows in the right place. 4. Send post request to manually setup an illegal meeting, see whether it is handled properly.	Validation Test Defect Test Unit Test	C
Group Creation Test	User should have the ability to create a group so that it is easier from them to create a meeting with the targeted group. The group information needs to be stored in the database, which	1.Use commands in MongoDB to insert a new document that contains a set of users as a group into Group collection. 2.Change one of the user to a nonexistent user, and see if this illegal operation is handled properly. 3.Use user-interface	Validation Test Defect Test	X

	should be tested on back-end only and interface between front-end and back-end just as we test the configure meeting.	(front-end) to create a group and then check if this newly created group shows under the current groups.		
Scheduler Test	We will test if a user can schedule a meeting by submitting an online form. The form should contains some information provided by the invitee and when submitted, there should be a record of it in DB and it will be shown in invitee's dashboard.	1. Fill up the form with invalid input (e.g. wrong email format) 2. Check if the information shown in the form matches what in DB 3. Check if submitted form is stored in DB 4. After submission, check if invitee's dashboard is updated accordingly.	Component Test Validation Test Defect Test	O
Sprint 2 User/Release Test	We will walk through the application as a user, who only interacts with the front-end web page of our application and we will try every function implemented in this sprint.	1. Register and login as a user 2. Create a group with several users 3. As a user, configure a meeting type and let another user to schedule a meeting with this meeting type	Validation Test User Test Release Test	Team

Sprint 3 (April 2nd - April 15th)

Task ID	Description	Sample Test Cases	Test Methods	Owner
SendEmail Test	Emails will be sent to verify registration, make invitation and remind users of	1. Register new test user and verify via email, check db to ensure new user persists. 2. Make a new invitation in the web app and check target recipient email inbox to verify	Validation Test, Defect Test	C

	meetings.	message and contents. 3. Make meeting with meeting time reminder threshold set to current or near to the current time and check for a reminder email.		
Meeting Edition UI Test	This interface allows users to modify existing meetings. It provide access to change a meeting's date, location or to cancel it.	1. Perform tests as a user working directly with the web app which modify an attribute or a combination of attributes in a selected meeting and verify proper behavior. 2. Perform tests as a user working directly with the web app which cancel meetings and verify proper behavior.	Component Test Validation Test, Defect Test	X
DbUpdate Test	After meetings have been rescheduled, the corresponding information in database will also be updated and the members of meetings should be notified.	1. Lookup meeting and its settings in db, run testcase to modify one or a combination of attributes and save, then query that meeting to verify changes are reflected. 2. Perform a similar set of operations but as a user working with the web app instead of JUnit tests and testing in Eclipse. Ensure changes are reflected in db and members of meeting have received notification.	Unit Test, Defect Test, Validation Test	X
~~HomePage~~ Welcome Page Test	The ~~homepage~~ welcome page (page before login) will provide a basic introduction of Meeting Manager	1. Navigate to web address hosting Meeting Manager web app and check that welcome page is there and displays correct contents properly.	Unit Test, Defect Test, Validation Test	O
Response View Test	Users can respond to a meeting request. Users can either accept or reject an invitation. After the date of	1. An invitation will be sent to a user's email box. We will test whether the user can respond to this request(i.e., accept it or reject it). 2. We will test whether the user can respond to a	Validation Test, Unit Test, Defect Test	D

	a invitation, user should be no longer able to modify it.	invitation who's date has already past.		
Response in Db Test	After the User has respond to an invitation, database will record information that whether the user accept an invitation or not.	1. We will run MongoDB and modify the meetings which are already saved in it. 2. After the user has responded to the invitation, we will test whether the data is saved in database.	Validation Test Unit Test	D
Invitation Test	Users can send invitations to whom the user want to meet with. This can be done via email or app notification (or both).	1. We will log in as a user and send an invitation to another user. Then we will log in the other user and check whether there is an email or notification. 2. We will log in as a user and send many invitations to one user. Then we will log into that user account to check whether he has received X invitations. 3. We will log in as a user and send many invitations to many users. Then we will log into those users and check whether each of them has received Xi invitations.	Validation Test Unit Test Component Test	C
Final Product User/Release Test	Users can use the Meeting Manager web application and perform or interact with all specified functionalities from Project Plan	1. This test will consist of a demonstration where a new user registers with our web application and executes all specified functionalities.	Release Test, User Test, Validation Test	Team